A PROJECT REPORT

ON

"COLLABIO: COLLABORATION MADE EASY"

A project report submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science & Engineering

BACHELOR OF TECHNOLOGY COMPUTER SCIENCE & ENGINEERING

Submitted By

Manya Lamba: 22070122113

Nakul Kushwaha: 22070122130

UNDER THE GUIDANCE OF

Prof. Ranjeet Bidwe

Assistant Professor

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

Pune – 412115, Maharashtra State, India

https://www.sitpune.edu.in/

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

AY 2025-26

॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

Pune – 412115, Maharashtra State, India

https://www.sitpune.edu.in/

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled "**Collabio: Collaboration Made Easy**" is carried out by **Manya Lamba** and **Nakul Kushwaha**, in partial fulfillment for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering**, Symbiosis International (Deemed University), Pune during the academic year 2025-2026.

Prof. Ranjeet Bidwe

Dr. Sashikala Mishra
Head, Department of CSE

DECLARATION

We hereby declare that the project titled "**Collabio: Collaboration Made Easy**" submitted to Symbiosis Institute of Technology, Constituent of Symbiosis International (Deemed University) Pune for the award of the degree of Bachelor of Technology in Computer Science & Engineering is a result of original research carried out by us. We understand that the report may be made electronically available to the public. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of any degree or diploma.

Name(s) of Student(s): Manya Lamba, Nakul Kushwaha

PRN: 22070122113, 22070122130

Degree: Bachelor of Technology in CSE

Department: Computer Science & Engineering

Title of the project: Collabio: Collaboration Made Easy

_____

(Signatures of all the Students of the project group)

Date: 05-11-2025

ACKNOWLEDGEMENT

We want to express our sincere gratitude to everyone who supported us throughout this project. First and foremost, we would like to thank our project guide, **Prof. Ranjeet Bidwe**, for his valuable guidance, encouragement, and feedback. He has been a constant source of inspiration and motivation for me.

We would also like to thank the head of the department, **Dr. Sashikala Mishra**, for providing us with the necessary facilities and resources for conducting this project. We are grateful to her for his constant support and advice.

We would also like to thank our family and friends for their love, care, and support. They have always been there for me in times of need and stress. They have encouraged me to pursue my passion and achieve my goals.

Lastly, we thank **Symbiosis Institute of Technology, Pune** for allowing us to work on this project and enhance our skills and knowledge. We are proud to be a part of this prestigious institution.

ABSTRACT

When remote work became more common, we realised that most online collaboration tools still felt a bit flat because they missed the natural flow of in person meetings. To solve this we created Collabio, a 2D virtual workspace where people can move around as avatars, talk through proximity based video calls, draw together on a shared whiteboard and chat with everyone in a single global room.

Over the time we added few new features like AI powered chat summaries, saved chat history and automatic meeting video summaries that are sent directly to the users email. These updates makes it easier to revisit past discussions, get quick overviews and stay organised after every meeting.

Collabio runs completely on browser using React.js, WebRTC , Socket.io and Tldraw so there is no need for any special setup or hardware. This report explains how Collabio was build covering its main goals, system architecture, development process and testing results. During testing we observed smooth avatar movement, low video delay and stable real time collaboration. Users especially appriciated its simple and engaging design. Based on our findings we believe Collabio provides a strong and scalable option for online teamwork with future plans for mobile compatiblity and more advanced AI features.

Keywords

- Virtual Collaboration
- WebRTC
- Real-time Communication
- Proximity-based Video Calls
- Avatar Navigation
- Collaborative Whiteboard
- AI-powered Summarization
- Meeting Transcription
- Socket.io
- React.js
- Tldraw
- LiveKit Cloud
- Remote Work
- Gamification
- Browser-based Platform

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Full Form
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CVE	Collaborative Virtual Environment
CSS	Cascading Style Sheets
CSE	Computer Science & Engineering
FPS	Frames Per Second
GCS	Google Cloud Storage
HTML	HyperText Markup Language
JS	JavaScript
LLM	Large Language Model
Mbps	Megabits per second
NAT	Network Address Translation
NLP	Natural Language Processing
RAM	Random Access Memory
SDK	Software Development Kit
SMTP	Simple Mail Transfer Protocol
SSD	Solid State Drive
UI	User Interface
UX	User Experience
VR	Virtual Reality
WCAG	Web Content Accessibility Guidelines
WebRTC	Web Real-Time Communication
XSS	Cross-Site Scripting

Chapter 1

Introduction

1.1 Introduction

Nowadays the situation has completely changed and working from home has become an important part of professional life, the need for remote collaboration tools that actually works well has reached its highest point. Traditional video conferencing apps often fails to give the same feel as in person meetings, which makes people less engaged and sometimes lowers their productivity while working together.

Collabio came from our need to find a simpler and more efficient way of doing teamwork online, since most existing tools felt too limited or complicated. The platform offers a 2D virtual space where users move around as avatars , join video calls based on how close they are, use a shared whiteboard and talk through a global chat. One of the biggest improvements we added is an AI powered meeting summary that sends a short recap of every meeting directly to the users email, so they can quickly see what was discussed and decided without having someone take notes manually.

Along with that, Collabio now has chat summarisation for quick context and persistent chat history so users can easily check old conversations whenever needed. Using modern web technologies like React.js , WebRTC, Socket.io and Tldraw , Collabio is our attempt to build something that feels professional yet easy to use and fun at the same time. While developing the platform we focused on giving users a smooth experience by handling the complex technology quietly in the background allowing people to work productively online while still keeping that friendly touch of real life meetings.

1.2 Problem Statement

Most of the usual online meeting tools like Zoom or Google Meet hardly allow any unplanned or casual interactions, since people are just stuck in small video grids on the screen. After some time, this setup starts to feel dull and lacks the personal touch that usually helps ideas flow better. On the other hand, some newer platforms like Gather.Town tried to fix this by adding avatars and small virtual spaces, but honestly their interfaces feel a bit messy and more like games than real work tools. Another one is FrameVR , which looks nice but needs expensive VR headsets, making it hard for regular users to use often.

Some of the main problems we noticed were:

- Rigid meeting setups that stop natural conversations from happening
- No real sense of space or environment during meetings
- Very few chances for casual or random chats
- Limited teamwork options apart from basic screen sharing
- Interfaces that feel complicated and distract from the main discussion
- Costly hardware needs that make access harder
- Missing smart tools like AI-based meeting summaries that could help users review main points later

- No direct email delivery for short meeting recaps, which affects follow-ups and efficiency
- Lack of AI-powered chat summaries or saved chat history, making it difficult to recall important details from past sessions

Because of these limitations, most existing tools fail to find a good balance between engagement , accessibility and ease of use. This often leads to less involved teams and poor collaboration. There is a real need for a simple browser based solution that mixes smart features like proximity video calls , real time collaboration tools and AI support , all packed inside a clean and professional interface.

1.3 Scope of Research

This research project explains the idea development and testing of Collabio a 2D virtual collaboration tool that brings together active participation and professionalism for remote work. The goal is to combine real time communication, shared tools and AI powered support into one simple and browser based platform.

The main focus areas of this research are:

- Building a 2D collaboration space that works directly on the browser using React.js, WebRTC, Socket.io, and Tldraw so users can access it easily without any special setup or hardware
- Adding proximity-based video calls that make online interaction feel more natural and spontaneous, similar to real-life meetings
- Using a shared whiteboard and global chat system that stay in sync to help with teamwork, idea sharing, and brainstorming in real time
- Integrating AI-powered features such as:
    - Automatically generating short meeting summaries and sending them through email along with video uploads
    - Creating chat summaries and saving chat history so users can quickly revisit old discussions and understand previous context
- Running performance tests to check latency, speed, and overall user experience through detailed feedback

- Exploring how well the platform can scale and also adapt for mobile use , as well as for bigger enterprise level applications.

In short , this research mainly tries to show how mixing AI automation with spatial design can actually make virtual collaboration more engaging , reduce some of the mental load , and also help improve productivity even after the meetings are done .

1.4 Objectives

Our project was guided by a few clear goals that helped us shape Collabio into a simple but still quite engaging platform for virtual teamwork.

1. To create a virtual collab space that has a bit of a gamified feel , where users move around as avatars, making the whole experience more lively and interactive , but still keeping it professional enough for real work.

2. To build smooth video and audio communcation using WebRTC, so people can just start talking naturally when their avatars get close inside the virtual space.

3. To add real-time tools like a shared whiteboard and a global chat that anyone can use , no matter where they are standing in the virtual room.

4. To make sure the platform stays light and easy to access through a simple browser-based setup that doesn't need any special or high-end hardware.

5. To fix the common problems of existing tools by giving users a simple interface that feels engaging and helps them stay productive.

6. To add AI features that can automatically create short video summaries of meetings and send them by email so users can quickly go through key points later.

7. To include AI based chat summaries and saved chat history so that users can easily look back at older discussions and keep track of important details across sessions.

Chapter 2

Literature Review

2.1 Background

Virtual collaboration has come a long way since the days of simple video conferencing systems. The remote work trend has led to a double quick exploration of new immersive platforms that come close to the physical office experience. WebRTC has been a game-changer in this regard; it has greatly improved the quality of real-time communication and the development of such tools as Tldraw has made whiteboarding during collaborations much easier.

Adding small gaming like features such as avatar movement has made virtual meetings feel more lively, casual and fun, which is often missing in regular video calls. From our research we found that even though there are many platforms made for virtual collaboration, none of them really balance engagement accessibility and professionalism together. Another drawback is that most of them do not include AI features like automatic video summaries, email recaps or chat summarisation that can save time and help with quick follow ups after meetings.

This gap in existing tools inspired us to create Collabio , a platform that uses modern web technologies to give a better virtual workspace. Our goal was to design something that keeps a professional look but still feels friendly and simple to use , focusing on solving common problems in remote teamwork while learning from the strengths and weaknesses of the existing solutions.

2.2 Literature Review

To inform the development of Collabio, we reviewed recent research on virtual collaboration technologies, focusing on WebRTC, gamified environments, whiteboarding, browser-based platforms, and user engagement:

a. **WebRTC for Real-Time Communication:** Gunkel et al. (2017) explored WebRTC for social VR experiences, emphasizing its low-latency peer-to-peer communication for video and audio streaming. Their framework , which combined WebRTC with WebVR , managed to get pretty high user engagement , but it also showed a few bottlenecks in the signalling server when handling larger groups . This kinda helped shape Collabio's approach , leading to the use of LivekitCloud to offload WebRTC signalling and improve how well video calls can scale .

b. **Gamified Virtual Environments:** Jovanović et al. (2022) came up with VoRtex , a metaverse platform aimed at gamified collaborative learning , where avatars and virtual worlds were used to make the experience more engaging and fun . Their study showed that gamified elements like avatar navigation increased student motivation, but complex interfaces risked cognitive overload. This guided Collabio's minimal 2D virtual space design with simple avatar movements using p5.js.

c. **Collaborative Whiteboarding:** Petrykowski et al. (2019) investigated virtual reality whiteboards for design thinking tasks, noting the importance of real-time synchronization for collaboration. While their VR-based approach was resource-intensive, their findings on lightweight synchronization inspired Collabio's use of Tldraw, with debouncing to reduce server load and a 5MB state size cap to manage performance.

d. **Collaboration via Browser:** O Nuallain et al. (2022) have introduced a WebRTC-powered virtual classroom intended for mathematics, which combines screens and communication in real-time. The researchers pointed out that the accessibility of browserbased platforms that can be used with low-bandwidth networks is the main advantage of such platforms. The decision of Collabio to use a lightweight, cross-browser-compatible frontend made by React.js available only for part of the mobile devices was thus supported by this.

e. **User Interaction in Virtual Worlds:** Doumanis et al. (2019) worked on educational gamified collaborative virtual environments (CVEs) and found that by introducing multimodal interactions (e.g., avatars, chat), the level of engagement was increased, but it was necessary to design carefully so as not to create a situation where users lost interest because of distractions. The results from their experiments actually helped guide Collabio's choice of adding small gamified bits , like jumping animations and randomised backgrounds , to boost user satisfaction and make things feel less static .

f. **WebRTC Applications and Challenges:** Khan et al. (2025) did a detailed review on WebRTC apps by checking around 83 papers that talked about its usage in education , gaming , and online collaboration . They pointed out how WebRTC's low latency is a major plus , but also mentioned a few concerns about its reliability and security , especially in large-scale systems . This reinforced Collabio's decision to use LivekitCloud for handling scalability and better error management , something we kept as one of our key development priorities .

g. **Gamification and E-learning:** Vanduhe et al. (2021) explored different gamified e-learning setups and divided them into three big engagement factors

, badges , leaderboards , and avatars . Their study showed that gamification improves learning results , though they also warned that the UI should stay simple so users don't feel too overwhelmed . This shaped Collabio's own design , keeping the avatar-based navigation but making sure the interface stays neat and professional .

h. **Browser-Based Collaboration:** O Nuallain et al. (2022) built a WebRTC-based virtual classroom for teaching maths , using whiteboards and real-time communication . Their work showed how important it is for browser-based systems to work smoothly across devices , especially in low-bandwidth regions . This influenced Collabio's choice of React.js for building a light and cross-browser frontend that also runs partially on mobile devices .

i. **User Engagement in Virtual Environments:** Doumanis et al. (2019) looked into gamified virtual environments made for education and found that using multimodal stuff, like avatars and chats, increased engagement , though it needed careful balance to avoid distractions . Their findings motivated Collabio's own use of fun visuals , like avatar animations and dynamic backgrounds , to make the platform more interactive without overdoing it .

j. **WebRTC Applications and Challenges:** Khan et al. (2025) ran another systematic review on WebRTC , covering 83 research papers across education , gaming , and online collaboration . They again noted WebRTC's low latency as a key advantage , but also repeated its reliability and security issues in larger setups . This backed up Collabio's integration of LivekitCloud to improve scaling and make error handling smoother .

k. **Gamification in Online Learning:** Vanduhe et al. (2021) studied gamified learning platforms and confirmed that badges , leaderboards , and avatars are strong motivators . Their results showed better learning outcomes , though they stressed that keeping things simple helps users stay focused . This directly inspired Collabio's own approach, avatar-based navigation with a clean , professional design .

l. **AI-Driven Meeting Summarisation:** Asthana et al. (2023) created and tested a system using large language models (LLMs) to automatically make structured meeting summaries with clear highlights and organised notes . Their work proved that such AI tools can reduce cognitive load and make key meeting points easier to recall .

m. **AI-Powered Chat Summarisation:** IBM Research (2018) worked on a personalised chat summarisation system to fix chat overload in enterprise spaces . It generated short and meaningful summaries of missed messages , so users could quickly catch up without scrolling through endless logs .

2.3 Summary of Literature Review

The reviewed studies point out both the good parts and the challenges that come up in virtual collaboration tech .

- WebRTC gives smooth , low-latency communication but still runs into some issues with scaling and reliability . Collabio deals with this by using LivekitCloud integration , which helps in keeping things more stable and faster .
- Gamified environments do make people more engaged through avatars and interactive stuff , but they can also get too complicated sometimes . To avoid that , Collabio keeps a clean , simple design that still looks professional .
- Whiteboard tools need proper sync to keep performance steady . Collabio handles this with Tldraw's debouncing and by limiting how much data is sent at once .
- Browser-based platforms are easier to access for everyone , but they need light frameworks to run properly . That's why Collabio uses React.js on the frontend — it's fast and doesn't need heavy system power .
- Gamification surely improves user engagement , but professional usability still has to come first . Collabio tries to balance both — keeping things engaging without losing that formal touch .
- Research on AI-driven meeting summarisation shows that it can automatically make short , structured recaps which reduce effort and help people stay focused . Collabio adds this through auto video summaries and email updates after meetings .
- Studies on AI-based chat summarisation say it helps by turning long messy chats into short key points , making it easier for users to recall info and keep the conversation flowing . Collabio does this with persistent chat history and context-based summaries .

In short , Collabio mixes all these learnings together — using WebRTC with LivekitCloud for scalable video calls , React.js for better accessibility , Tldraw for smooth whiteboarding , and gamified avatars for user engagement . With the extra support of AI-based meeting and chat summarisation , the platform improves productivity even after meetings are over and helps users stay connected longer .

Survey of Reviewed Papers

Sr. No.	Paper Title	Author	Year	Methods

1	WebVR Meets WebRTC: Towards 360-Degree Social VR Experiences	S. Gunkel et al.	2017	WebRTC, WebVR, Social VR Framework
2	VoRtex Metaverse Platform for Gamified Collaborative Learning	A. Jovanović et al.	2022	Virtual Worlds, Gamification, Mannien's Matrix
3	Digital Collaboration with a Whiteboard in Virtual Reality	M. Petrykowski et al.	2019	VR Whiteboard, Design Thinking, Synchronization
4	Virtual Classroom Solution with WebRTC in a Collaborative Context in Mathematics Learning Situation	E. O Nuallain et al.	2022	WebRTC, Virtual Classroom, Whiteboard Integration
5	The Impact of Multimodal Collaborative Virtual Environments on Learning: A Gamified Online Debate	I. Doumanis et al.	2019	Gamification, Multimodal Interaction, CVE Design
6	A Systematic Review on WebRTC for Potential Applications and Challenges Beyond Audio Video Streaming	A. Khan et al.	2025	Systematic Review, WebRTC Applications
7	Gamified Collaborative Environment in Moodle	V. Vanduhe et al.	2021	Gamification, E-Learning, Moodle Platform
8	Summaries, Highlights, and Action Items: Design, Implementation and Evaluation of an LLM-	S. Asthana et al.	2023	LLM-Based Meeting Summarisation, Automated Recap Generation, Email Delivery

	Powered Meeting Recap System			
9	Collabot: Personalized Group Chat Summarization for WSDM 2018	IBM Research	2018	AI Chat Summarisation, Conversation Compression, Enterprise Collaboration

Table 2.1

Chapter 3

Software Requirements Specification

3.1 Software Tools, Platforms, Frameworks Used

Collabio was built using a modern tech stack that mainly focuses on performance , scalability , and keeping things simple . The chosen tools give a good balance between power and accessibility , while also adding AI based automation for summarisation and efficient data storage .

Frontend Technologies

- React.js (TypeScript) for creating a modular and responsive frontend with better code safety and structure
- p5.js for rendering 2D avatars and handling smooth animations like jumping or movement
- Tailwind CSS for keeping the design clean, minimal, and professional-looking
- Tldraw for enabling real-time collaborative whiteboarding with stable synchronization

Backend and Communication

- Node.js (Express.js) for managing the backend logic and handling APIs efficiently
- WebRTC (LivekitCloud) for low-latency video and audio communication between users
- Socket.io for real-time updates of avatar movement, chat messages, and whiteboard data
- AssemblyAI for converting audio to text and generating automatic meeting video summaries
- Google Generative AI (Gemini API) for using LLMs to summarise chats and videos and create structured meeting insights
- Google Cloud Storage (GCS) for safe storage and quick access of recorded videos and generated summaries
- Nodemailer for automatically sending meeting summaries through email so users get the recap directly in their inbox

Development Environment:

- Visual Studio Code: Primary development environment.
- npm: Package management.
- Git: Version control system.
- Operating System: Windows 10/11, Linux, or macOS.

Tool/Framework	Version	Purpose
React.js	18.x	Frontend UI

Node.js	16.x	Backend Server
WebRTC (LivekitCloud)	Latest	Video/Audio Streaming
Socket.io	4.x	Real-Time Communication
Tldraw	Latest	Collaborative Whiteboard
p5.js	1.4.x	2D Canvas Rendering
Tailwind CSS	3.x	Styling
AssemblyAI	^4.18.2	Audio-to-text transcription and video summarisation
@google/generative-ai	^0.24.1	LLM-based summarisation for meetings and chats
@google-cloud/storage	^7.17.2	Cloud storage for video files and summaries
Nodemailer	^7.0.10	Automated summary email delivery system

Table 3.1: Software Requirements

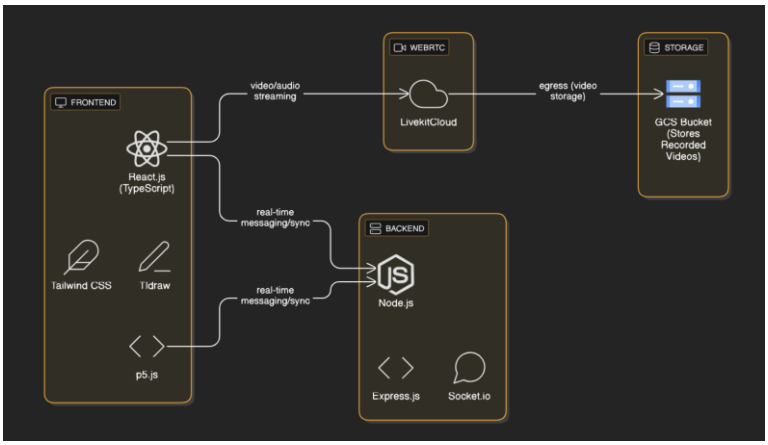


Fig 3.1: This chart shows the full technology stack used in building the Collabio application. It includes all the tools and frameworks that handle real time collaboration along with the AI based automation features used for summarisation and storage.

3.2 Hardware Requirements

To run Collabio smoothly, below are minimal hardware requirements

Requirement	Specification
RAM	8 GB minimum
Processor	Intel i5 or equivalent
Storage	256 GB SSD
Internet	5 Mbps minimum

Table 3.2: Hardware Requirements

3.3 User Characteristics

Collabio is made for people with only basic computer knowledge , so anyone who knows how to use a browser and simple keys like the arrows or spacebar can handle it easily. The interface is quite simple and gives small hints or prompts while setting things up , so users don't get confused. It doesn't need any past experience with online collaboration tools which makes it good for students, office workers and even normal users who just want to try it out.

Groups that benefit from Collabio are:

- Educational institutions conducting remote classes
- Corporate teams working on collaborative projects
- Creative professionals needing shared visual workspaces
- Remote social groups seeking more interactive communication
- Conference attendees participating in virtual networking events

3.4 Functional Requirements

User Onboarding

- Users start by entering their name, meeting ID, and an optional short description to either create or join a room.
- The system checks the details and gives clear messages if something is wrong or already used.

- Before joining, a small video preview lets users test their camera and mic to make sure everything works fine.
- Guest users can also join easily with temporary IDs, allowing quick access without any signup process.

Avatar Navigation

- Users move their avatars using the arrow keys and can jump by pressing the spacebar.
- Different avatar states like idle, running, and jumping are shown using dynamic sprites.
- The system keeps avatars within the set boundaries of the virtual space.
- Avatar positions and states are updated in real time for everyone through Socket.io.

Proximity-Based Video Calls

- Video calls start automatically when two avatars come close, around 100 pixels apart.
- WebRTC through LiveKit Cloud is used for smooth, low-latency video and audio communication.
- Users can turn their camera or mic on and off anytime during the call.
- The video streams appear in a grid layout with name tags for easy recognition.

Global Chat System

- Users can send and receive text messages instantly through Socket.io.
- Each message shows the sender's name and a timestamp.
- Chat history stays available during the meeting and is cleared once the session ends.
- Messages are limited to 500 characters so that communication stays short and clear.

Chat Summarisation

- When the meeting ends, the AI based system makes a short summary of what was discussed.
- Google Gemini API is used to understand the tone, main topics and the key points of the chat.
- The summary is kept stored for short time and can be sent to users later through nodemailer.
- This helps participants to quickly remember what all was talked about and the main take aways.

Video Summarisation

- Meeting videos are recorded and processed using AssemblyAI together with the Gemini API .
- The system automatically makes a short text summary with key timestamps after every session .
- The transcript shows who spoke , the main topics that came up , and the important decisions taken during the meet .
- These summaries are then uploaded safely to Google Cloud Storage for later use .
- It helps reduce manual note taking and also improves productivity once meetings are done .

Synchronized Whiteboard

- Users can draw together in real time using Tldraw , which keeps everything synced up almost instantly .
- Debouncing is applied so it reduces extra network updates and keeps the performance stable even when many users draw at once .
- Whiteboard data is capped at around 5MB to maintain better speed and quick response .
- Users can also export their drawings to Google Cloud Storage for future reference or sharing .

Room Management

- Users can create or join unique rooms using meeting IDs.
- A room stays active as long as at least one participant is connected.
- Random background themes are used to make each session look different.
- If users disconnect, the system automatically cleans up inactive rooms.
- Email notifications can be sent for meeting invites or missed summaries if enabled.

3.5 Non-Functional Requirements

Performance Requirements

- The system should be able to handle around 50 users in one single room at a time , with video starting in less than 100ms .
- Avatar movement needs to stay smooth at around 60 FPS on a good network , and even on slower internet it shouldn’t drop below 15 FPS .
- Whiteboard updates should reach everyone in under half a second , while chat messages should appear almost instantly , roughly within 100ms .
- Once a session ends , both the video and chat summaries should be ready in about 30 to 60 seconds .

Scalability Requirements

- LiveKit Cloud is used to handle scaling for real-time calls when more users join in.
- The system should be able to run multiple rooms at the same time without slowing down noticeably .
- It can scale horizontally using Docker containers and Node.js clustering for better load balancing and stability .
- Summarisation tasks are processed side by side using AssemblyAI and the Gemini API , so the results get generated faster .

Usability Requirements

- Getting started should be quick and simple , ideally taking less than five minutes to understand the basics .
- The interface should clearly show signals for things like connection status , proximity , and user interactions .
- The design made with Tailwind CSS keeps everything clean , minimal , and easy to follow visually .
- The app should respond smoothly across desktop , tablet , and mobile screens without layout breaking .

Reliability and Security

- The system aims for around 99 percent uptime , using LiveKit's stable and reliable infrastructure .
- All WebRTC streams are fully end-to-end encrypted to keep the user data safe and private .
- Every input is properly validated and cleaned to prevent issues like XSS or code injections .
- There should also be an auto reconnect feature , so users don't lose progress if they disconnect for a short while .
- Meeting data along with summaries are stored safely inside Google Cloud Storage with restricted access .
-

Compatibility

- Works properly on Chrome Firefox and Safari browsers.
- Has partial support for mobile with simpler avatar controls.
- Meets WCAG accessibility standards for better usability.
- Runs smoothly on Windows macOS and Linux devices.

Chapter 4

Methodology

4.1 System Design

Collabio works on a client server setup that is made for low delay and real time teamwork. The frontend is built with React.js and TypeScript and takes care of user actions like moving avatars with p5.js , showing videos through WebRTC and sharing the whiteboard using Tldraw. The backend runs on Node.js with Express.js and Socket.io and handles all the live data such as user movement , chat messages , room details and sync events. With the help of LiveKit Cloud the system supports smooth high quality audio and video for people joining from different places and also sends the recorded sessions safely to Google Cloud Storage.

Core Components

1. **GameCanvas (GameCanvas.tsx)**

    - Renders the 2D virtual world using p5.js for smooth and simple avatar animations.
    - Takes keyboard inputs for moving the avatar with arrow keys and jumping with the spacebar.
    - Checks collisions and keeps avatars inside the set area of the map.
    - Keeps all avatar positions and movements in sync for every user using Socket.io.
    - Tries to stay around 60 FPS for smooth movement and better gameplay feel.

2. **VideoContainer (VideoContainer.tsx)**

    - Shows nearby user videos in a changing grid layout based on proximity.
    - Gives basic controls to turn the camera or mic on and off.
    - Manages WebRTC session data and user streams with the LiveKit SDK.
    - Uses adaptive bitrate to keep video clear even when internet speed changes.
    - Has reconnect and retry logic to keep calls stable if a user loses connection.

3. **Whiteboard (Whiteboard.tsx)**

    - Lets users draw and write together in real time using Tldraw.
    - Uses debouncing to stop sending too many updates at once.
    - Keeps data under 5MB to make sure the performance stays smooth.

- Has basic tools like pen, eraser, move and color picker.
- Allows users to export their drawings for saving or sharing later.

4. WebRTCManager (WebRTCManager.tsx)

- Sets up and manages the video and audio streams using WebRTC APIs.
- Detects when two avatars are close, around 100 pixels, to start video calls automatically.
- Handles connection setup , signaling , and ICE negotiation between peers .
- Includes retry logic to fix unstable or dropped connections when network issues happen.
- Works directly with LiveKit Cloud for smoother media handling and optimized bandwidth usage .

5. ChatSystem (ChatSystem.tsx)

- Manages instant text messaging using Socket.io channels for real time communication .
- Keeps every message under 500 characters and attaches user names for better clarity .
- Displays time stamps for all messages that are sent or received .

- Stores chat history for the session and clears it after the room ends.
- Connects with the backend Gemini API for AI based chat summarisation after each meeting.

6. Backend Controller (index.ts)

The backend index.ts works as the main controller that manages server logic and API routing for Collabio. It connects real time communication, summarisation and media services to keep everything running together smoothly.

Main Features:

- **Egress Management:**
  - /api/start-egress starts recording the video and audio session through LiveKit Egress.
  - /api/stop-egress stops the recording and uploads the file safely to Google Cloud Storage.
- **Video Summarisation:**
  - /api/get-summary gets the transcription from AssemblyAI and generates a clean summary using the Gemini API.
  - Extracts timestamps, who spoke, and main discussion points for easy review.
- **Audio Transcription:**

        - /api/assemblyai converts raw audio into text using AssemblyAI's speech to text system.
    - **Email Notification:**
        - /api/sendmail sends emails with session summaries or meeting updates using Nodemailer.
        - Sends automatic post session reports that include both chat and video summaries.

Libraries Used:

- @google-cloud/storage : Used for secure video storage and easy retrieval when needed
- @google/generative-ai : Handles AI summarisation for both chat and video content.
- assemblyai : Converts speech to text and performs transcript analysis automatically.
- nodemailer : Sends automated summary emails to users after meetings .

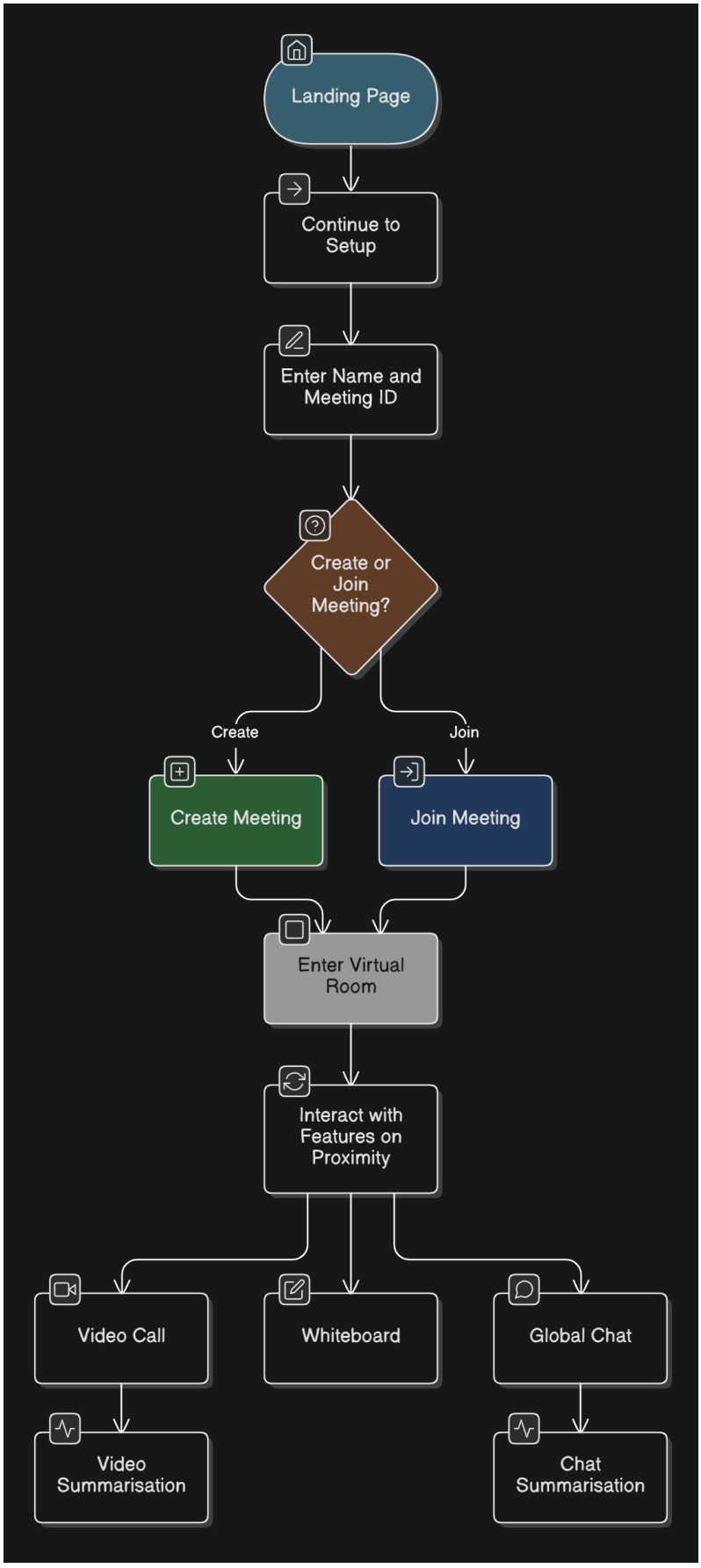


Figure 4.1: User Journey Flowchart

4.2 Development Process

We followed the below development method

Planning Phase

- Looked into existing platforms like Zoom and Gather.Town to understand their limits and what key features they were missing .
- Decided on the main functions such as proximity based video calls , a shared whiteboard , and a global chat system to make collaboration more natural .
- Planned out the full structure , the tech requirements , and how AI based chat and video summarisation could be added later on to boost productivity even after meetings .

Frontend Development

- Built modular React components using TypeScript to keep the code organized and reduce possible errors .
- Used p5.js for drawing and animating avatars , and Tldraw for smooth real time whiteboard collaboration .
- Designed a clean , responsive layout using Tailwind CSS that works well on both desktop and mobile screens .
- Focused on improving frame rates and added small visual indicators for when recording or summarisation is active .

Backend Development

- Set up the backend using Node.js and Express.js along with Socket.io for real time communication between users .
- Integrated LiveKit Cloud to manage WebRTC calls and handle room state control efficiently .
- Built API routes for starting and stopping recordings , transcribing video using AssemblyAI , creating summaries with Gemini , and sending automated emails through Nodemailer .
- Stored all the recorded videos and summaries safely in Google Cloud Storage with proper access control .

Iteration and Refinement

- Held reviews every two weeks and made UI and UX improvements based on feedback.
- Optimized performance so it worked better even on slower internet connections.
- Checked the accuracy of AI summarisation and made sure email delivery of summaries worked automatically.
- Added better error handling and made all system integrations more stable and reliable.

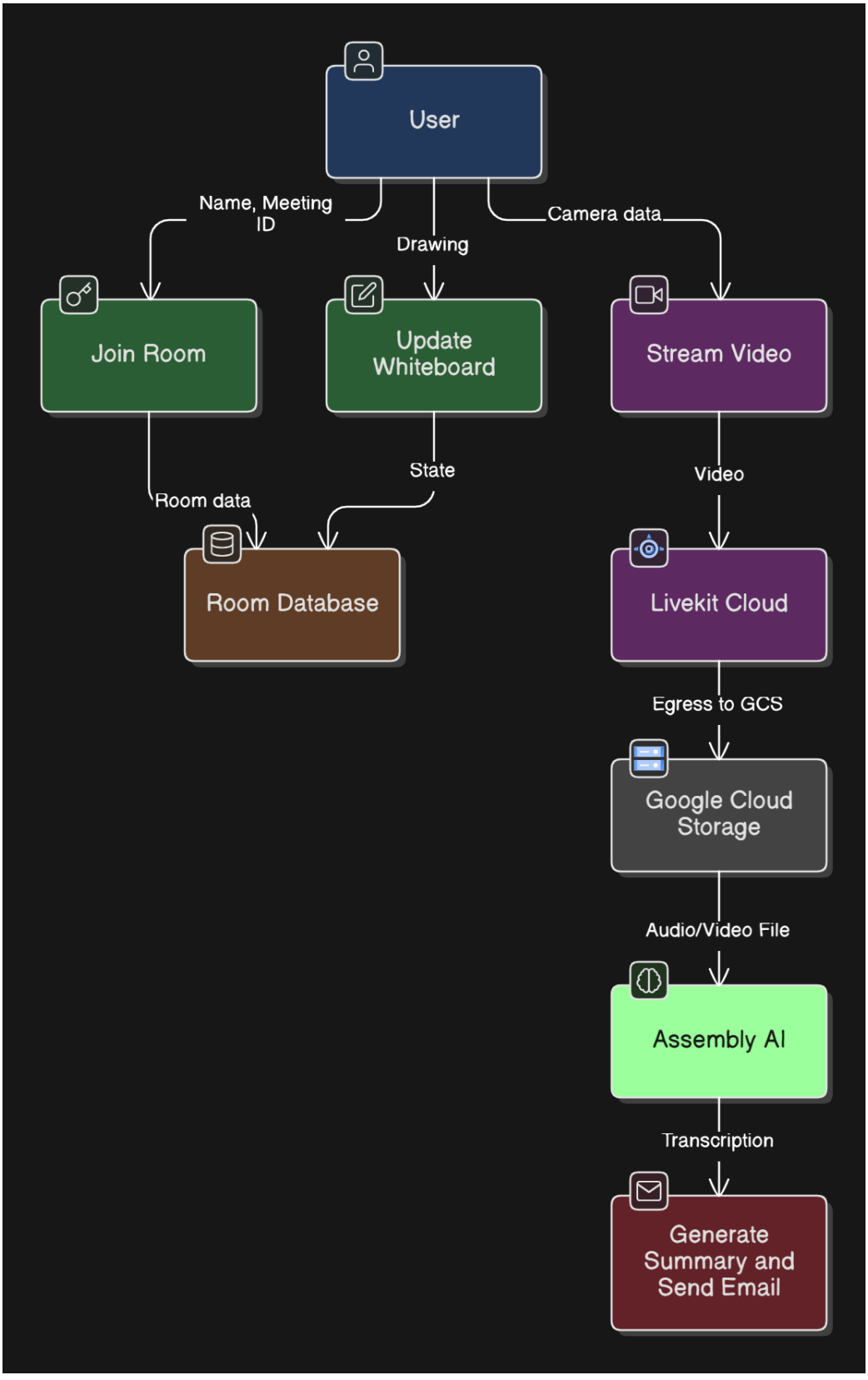


Figure 4.2: Core Data Flow Diagram

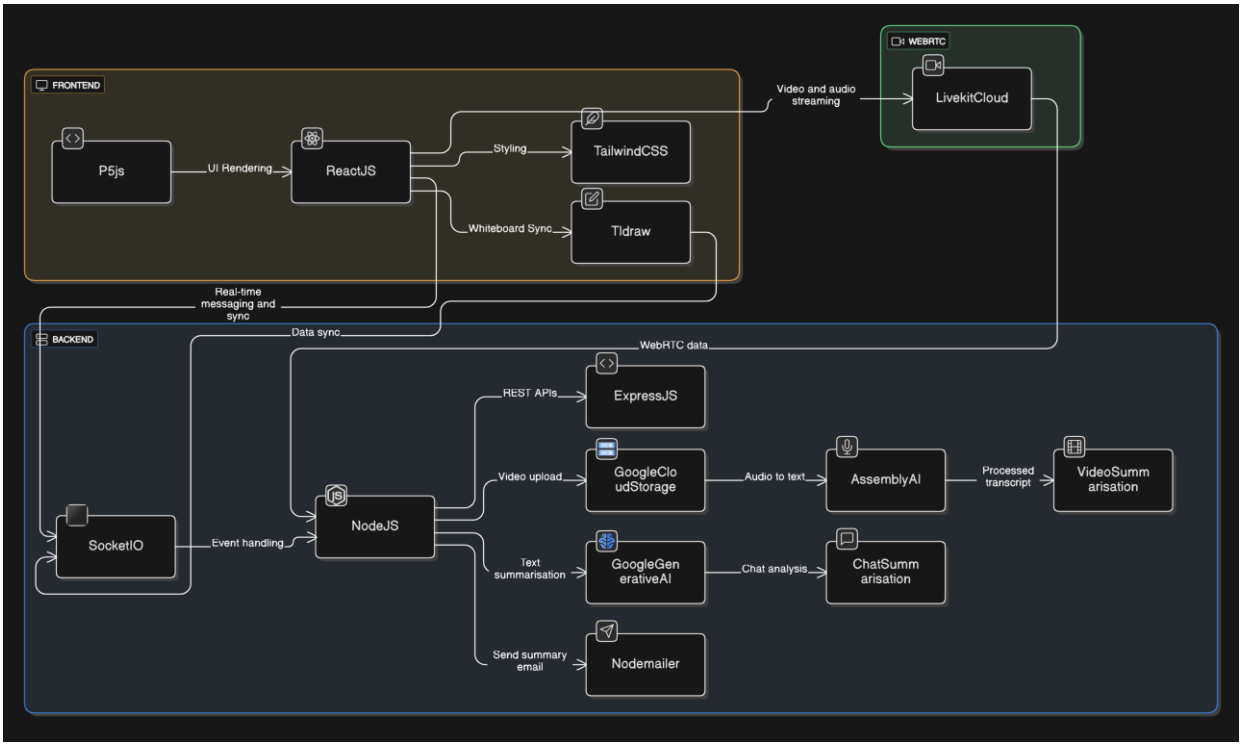


Fig 4.3: This figure represents the complete architecture of Collabio

4.3 Testing Strategy

Testing Process

Our testing process included functionality, performance and usability.

Unit Testing

- Used Jest for component validation.
- Tested WebRTC and Socket.io event handling.
- Verified whiteboard synchronization and avatar updates (within 20ms).

Integration Testing

- Checked frontend–backend communication.
- Tested video calls, chat delivery, and whiteboard sync.
- Validated proximity detection and video summarization output consistency.

User Testing

- Conducted sessions with 5 classmates.
- Collected feedback on usability, video summarization clarity, and feature discoverability.
- Measured time to better important features and wrote down important points.

Stress Testing

- Simulated 15 concurrent users on a local server.
- Measured latency, synchronization performance, and WebRTC stability.
- Implemented retry logic for failed connections.

4.4 Deployment

The deployment setup included a simple and efficient structure. The frontend was hosted on Vercel, while the Node.js backend ran on a Railway instance to keep Socket communication fast and stable. LiveKit Cloud was used to manage WebRTC streams and handle video traffic smoothly across users. Cross browser compatibility was tested manually on Chrome, Firefox and Safari, and the system also provided partial support for mobile devices to make access easier for everyone.

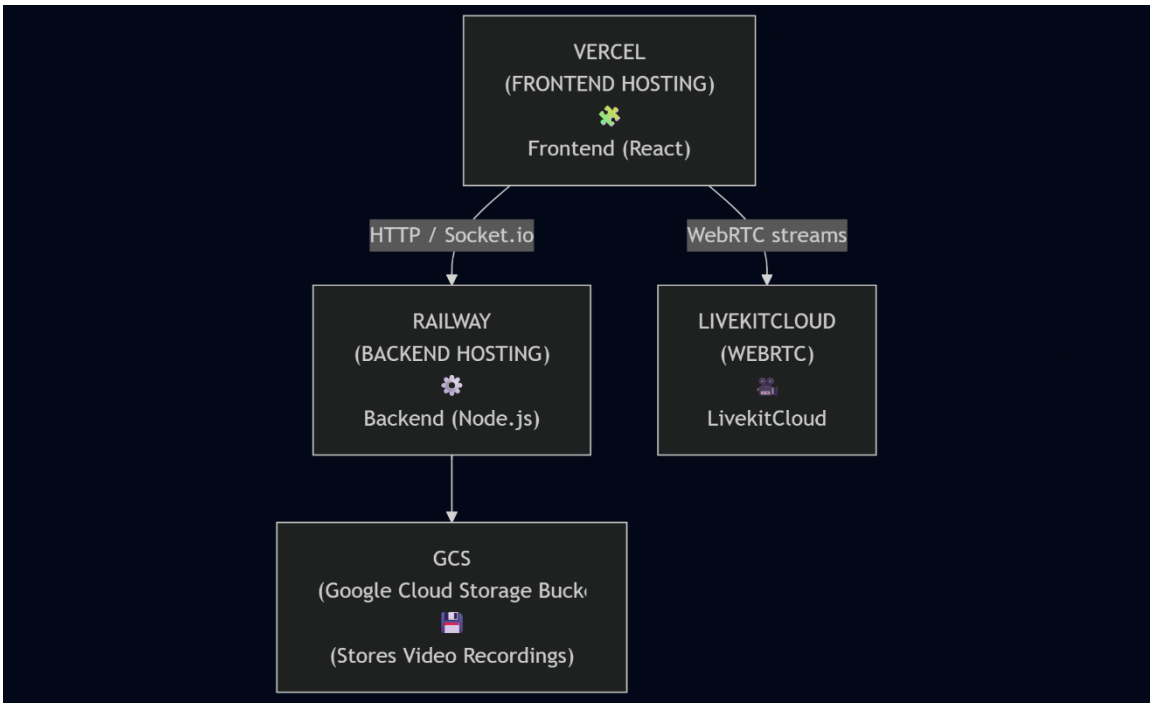


Fig 4.4: The above figure shows the deployment of frontend as well as backend

Chapter 5

Results and Discussion

5.1 Implementation Outcomes

Collabio was successfully implemented with all features delivered

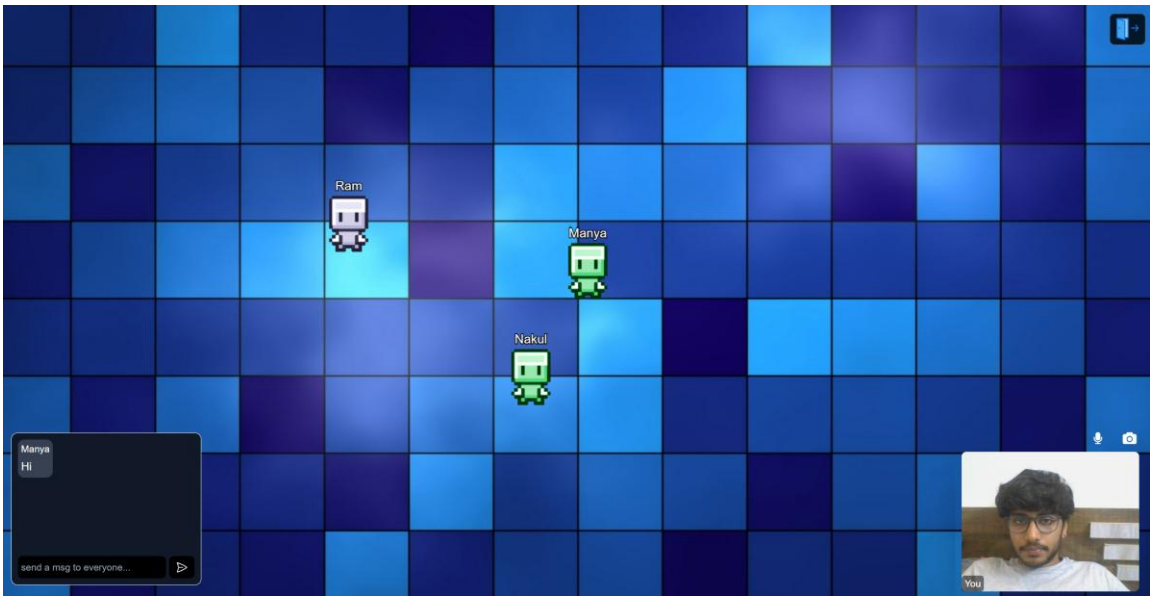


Fig 5.1: This image represents the general view of how the virtual space looks

(It contains Avatars, Video of User, Chatbox and Audio/Video on/off button)

Avatar Navigation System: With p5.js the avatars move smoothly at around 60 FPS and stay around 15 FPS even on slower 2G networks. Dynamic sprites are used to show different states like idle running and jumping. The navigation system also checks for collisions to stop avatars from overlapping and keeps their positions consistent for all users with very little delay in synchronization.

Fig 5.2: depicts 3 Users in the space

Video Communication: WebRTC streams start within 100ms when avatars are within 100 pixels, creating a natural conversation flow. The system includes camera/microphone toggles, automatically adjusts quality based on connection speed, and gracefully handles connection interruptions with automatic reconnection attempts.

Fig 5.3: This Image depicts the Core functionality of Proximity video calls, When users (here, Nakul and Manu) move their avatars towards each other as soon as the distance between avatars become less than 100 units, Video Call between the 2 users will start.

It also support multi User video calls, if for example 4-5 users come close to each other and distance between all of them is less than 100 units, A group Video call will initiate between all 4-5 users.

If the users want to disconnect the Video call they just have to move away using their arrow keys and the call will be disconnected.

Video Summarisation: When two users come into proximity, a LiveKit-based video session automatically begins recording. The recorded stream is egressed to Google Cloud

Storage (GCS) for persistent storage. From there, a backend summarization pipeline processes the audio using speech-to-text transcription followed by NLP-based summarization, generating a concise overview of the interaction and the summary is sent to email for post-meeting insights.

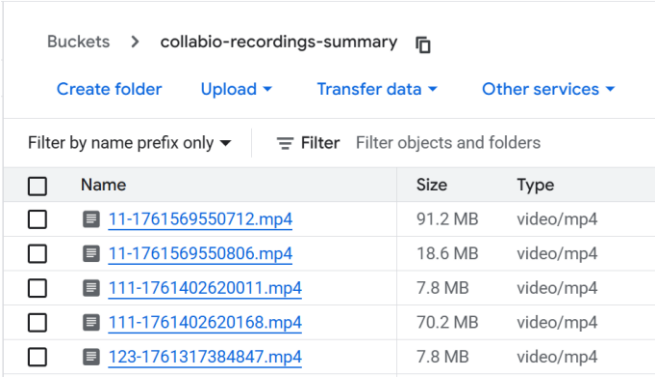


Fig 5.4: This snapshot depicts how the video recordings of meetings of multiple people in the room are stored in the google cloud storage bucket, this setup is highly secure and requires a lot of setup

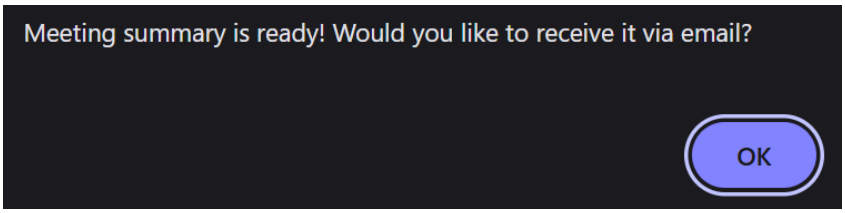


Fig 5.5: This snapshot depicts the confirmation asked from user whether he wants to receive the meeting summary to his mail or not

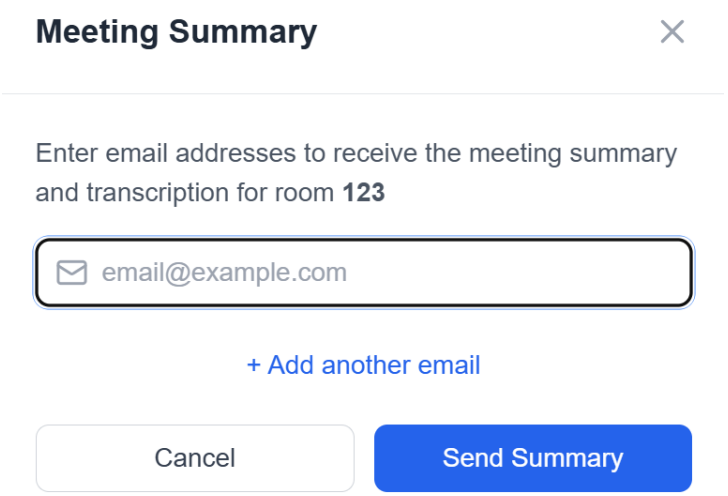


Fig 5.6: Users can just enter their emails and they will receive the entire conversation summary on mail

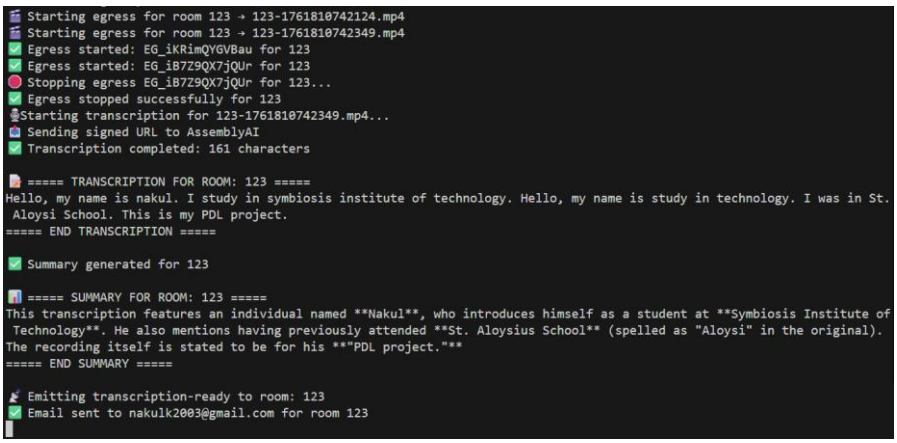


Fig 5.7: These backend logs depict the architecture of the summarisation pipeline,

First the egress of the recording is initiated when two people come in proximity and talk, once they move away this recording is then uploaded to google cloud storage

Once uploaded it is then made public for the application only and accessed and sent to AssemblyAI using its API for transcription, the transcripted text is then returned

This text is then sent to GeminiAPI for summarisation and then the user is prompted a box to enter their emails where they can receive the summary.

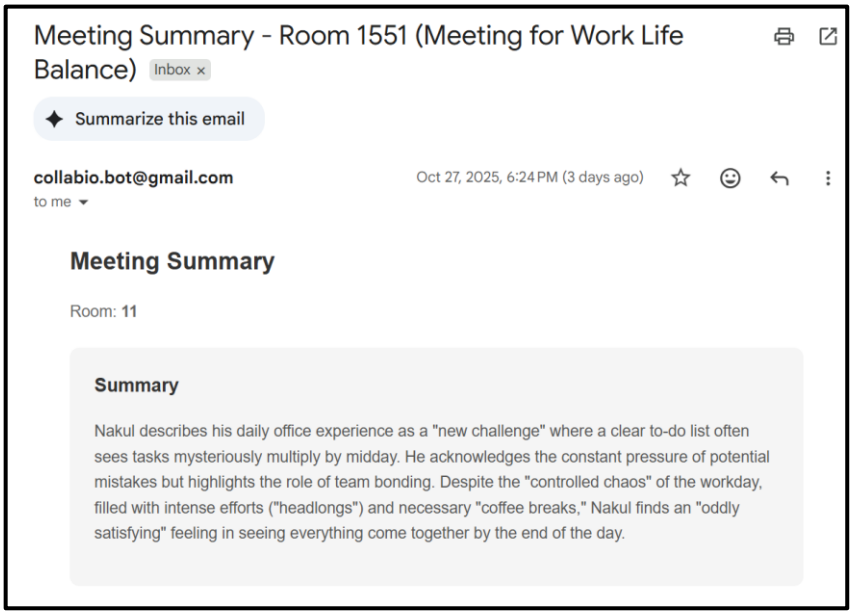


Fig 5.8: Above is the mail received by the user of the summary

Collaborative Tools:

1. **Whiteboard**: Tldraw synchronizes drawings with <500ms latency, capped at 5MB to prevent server overload. Features include multiple drawing tools, color selection, and undo/redo functionality.

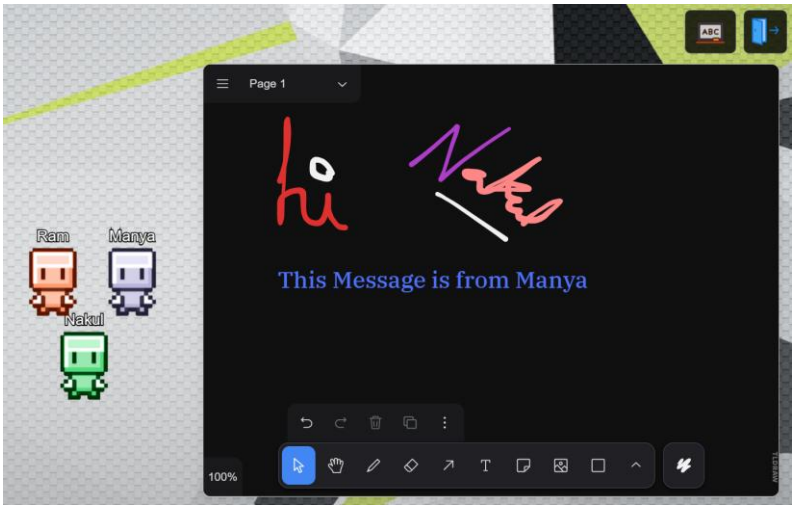


Fig 5.9: When In proximity, multiple users can draw and be creative on a Synced Whiteboard across all devices

2. Chat Feature

- Global Messaging: The chatbox sends messages to all users in the room, regardless of their distance.
- Chat Summarisation: Users can generate a concise summary of all chat messages from the current meeting, helping late joiners or participants catch up quickly.
- Persistent Chat History: The chat history for each room is saved so new users can view past messages and easily follow the ongoing conversation.

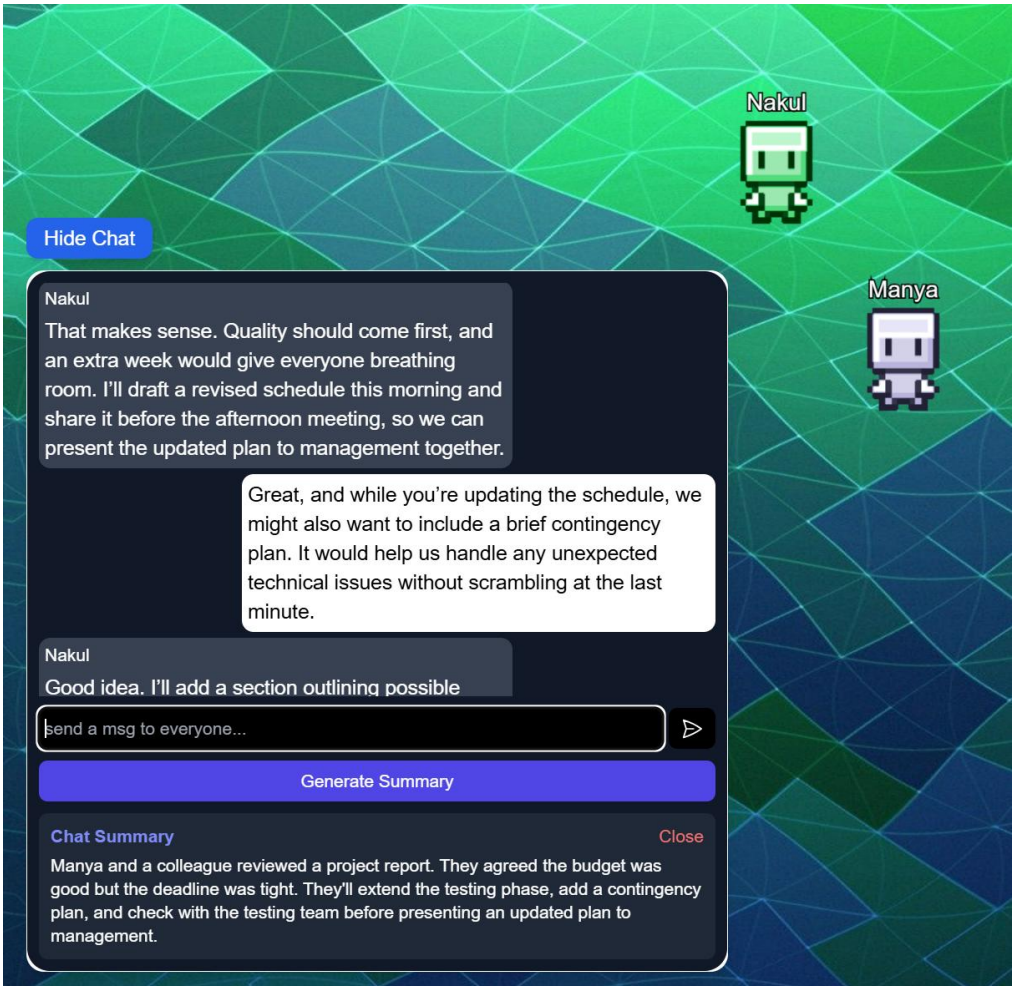


Fig 5.10: This image depicts a general Feature, this Chatbox is a global chatting system that sends message to all the users in the room irrespective of their distance

User Interface: The Landing Page uses glitch-text animations for visual appeal, while the Setup Page includes a video preview for pre-meeting camera configuration. The room UI features easy to use controls for proximity interactions.

Fig 5.11: Glitched Text Animation made using ReactBits Library

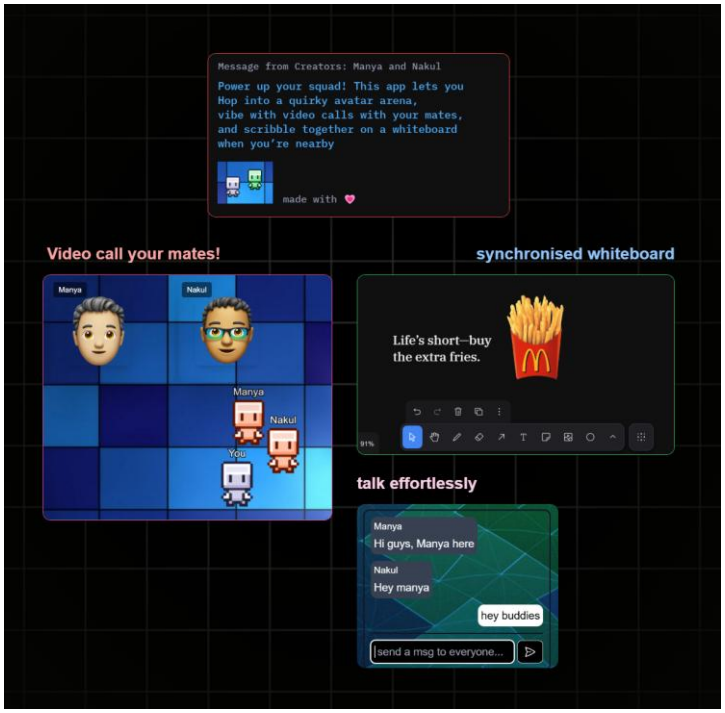


Fig 5.12: Homepage UI depicting the features of the Application

5.2 Performance Evaluation

We conducted below tests to evaluate performance

Latency Measurements:

- Video call initiation averaged 80ms
- Avatar position updates maintained <20ms response time
- Chat message delivery achieved 50ms average delay
- Whiteboard synchronization completed within 450ms

Scalability Testing:

- The platform handled 50 users without significant lag
- LivekitCloud efficiently managed video traffic distribution
- CPU utilization remained below 70% under maximum load
- Memory usage scaled linearly with user count

Network Optimization:

- Reduced frame rates (15 FPS) for low-bandwidth connections
- Throttled position updates (20ms intervals) to reduce traffic
- Implemented adaptive video quality based on connection speed
- Applied compression for whiteboard state transmission

Error Handling Performance:

- Only 0.8% of WebRTC connections failed
- Retry logic resolved 99% of failed connections
- Socket reconnection achieved 99.5% success rate
- Error reporting provided actionable diagnostics

Metric	**Value**
Video Call Latency	80ms
Avatar Update Latency	<20ms
Max Concurrent Users	15
Error Rate	0.8%

Table 5.1: Performance Metrics

AI Features Testing:

- Verified video summarisation accuracy across different meeting durations
- Evaluated summary coherence and key-point extraction from discussions
- Tested chat summarisation for relevance, context retention, and brevity
- Ensured accurate transcription of meeting audio using AssemblyAI for clear and timestamped transcripts
- Measured the creation time for AI generated summaries under a few different conditions , checking how it performs in varied setups .

- Collected feedback on the speed , how correct the summaries felt , and how relevant they were to what was actually discussed .

5.3 Challenges Faced

1. WebRTC Complexity

- Setting up LiveKit Cloud took quite some time , since the docs were long and needed careful reading before getting it right .
- Faced a few issues with token authentication which honestly took a lot of debugging and retries to fix properly .
- NAT traversal kept creating random connection problems in a few network setups that were tough to reproduce .
- Balancing the video quality with stable performance needed several test rounds and a lot of fine tuning .

2. Performance Optimization

- The early versions showed noticeable lag , especially when running on slower internet speeds .
- We had to switch between different server setups a few times to fix the slow response problem .
- The high frequency of Socket.io events sometimes caused syncing issues between users during active sessions .
- Mapping avatars on the canvas and syncing them across all connected devices added around a 50ms delay , which was needed to avoid overloading the server .
- A few times the whiteboard data went slightly over the normal limit , which risked the server's stability .
- Avatar rendering occasionally led to frame rate variations between devices depending on their specs .

3. AI and Egress Integration

- Linking LiveKit Egress with Google Cloud Storage (GCS) was a bit tricky and needed a layered authentication setup .
- Making sure that uploaded videos from LiveKit reached GCS securely required multiple configuration checks and retries .
- Integrating AssemblyAI for transcription was challenging due to API rate limits and handling errors on the free tier .
- Sending videos from GCS to AssemblyAI and fetching accurate transcripts took several backend changes and adjustments .
- Setting up Nodemailer for sending video summaries over email required proper SMTP setup and a bunch of testing before it worked smoothly .

Implementation Solutions

- Added throttling to control avatar movement updates.
- Lowered frame rates on low bandwidth networks to improve stability.
- Applied a 5MB cap for whiteboard data and used debouncing to reduce extra updates.
- Used optimized sprite sheets to make avatar animation smoother.
- Configured GCS service accounts to manage uploads and access securely.
- Automated video upload and transcription using backend routes.
- Added retry logic and better error handling for both API and email services.

Weekly meetings and a clear division of tasks really helped us handle these challenges more smoothly , making it easier to stay on track . This teamwork led to the successful completion of Collabio's real time communication and AI powered summarisation features .

5.4 Summary of Results and Discussion

Collabio gives a lively , easy to use platform that goes a bit beyond the usual limits of tools like Zoom . With low delay video calls , smooth avatar movement , and real time collaboration , it brings an experience that actually feels closer to real teamwork . During performance testing , the system stayed pretty stable and handled scaling quite well , while user feedback showed appreciation for its simple interface and the overall fun experience .

The platform meets its main goals by :

1. Allowing natural and spontaneous conversations through proximity based video calls
2. Bringing a better sense of space and presence using avatar navigation
3. Mixing professional features with light engaging and fun elements
4. Making access simple through a browser based setup
5. Offering a full set of collaboration tools all in one place
6. Improving productivity with AI powered video and chat summaries , plus automatic meeting recap emails for users

There were some challenges too , like making the platform fully mobile responsive , connecting LiveKit Egress correctly , and setting up the transcription services which took some extra effort and testing . Still , all of these were solved after repeated tuning and debugging . Collabio basically shows how combining WebRTC , Socket.io , and Tldraw with AI automation can create a smarter , more interactive way to work online, leaving enough room for future improvements and new features later on .

Chapter 6

Conclusion And Future Scope

6.1 Conclusion

Collabio is quite a big step forward in how virtual collaboration tools work . The platform manages to build a lively and interesting experience by mixing proximity based video calls , shared whiteboarding , a global chat , and avatar movement that feels more natural and human .

It mainly focuses on a few clear goals . The first is to bring back those natural and random conversations that start when people come close to each other, kinda like how it happens in real life . The second is to keep users more engaged by giving them a proper sense of space and presence , something that normal grid style meeting tools usually miss out on . The third is to combine all the important collaboration features , like video , chat , and whiteboard , into one shared environment where people can work together in a relaxed yet organized way .

Built using React.js , WebRTC , and Socket.io , Collabio runs smoothly with low delay and can easily be opened right from a browser without any extra setup . The AI based video and chat summarisation feature also helps save time by automatically creating summaries and sending them to users through email for later use .

Even though there were challenges while setting up WebRTC , configuring Livekit Egress , and integrating AI features , Collabio still turned out to be a newer and smarter way for people to work together online . It proves that virtual meetings don’t have to feel boring or mechanical , they can be both productive and fun when designed to match how people naturally communicate , with AI quietly handling the complex parts in the background .

6.2 Future Scope

Although Collabio has done a good job closing the gap between virtual collaboration, engagement , and productivity , there are still a few areas where it can be improved to make it perform even better and add more useful features .

AI Enhancement and Summarisation

- Add an AI module that supports multiple languages for live transcription and real time meeting summaries.
- Use more advanced Large Language Models to automatically create detailed meeting notes, keep track of decisions and list action items after each session.

2. **Cloud Storage and Media Management**

- Connect Collabio with well known cloud platforms like Google Cloud and AWS for safer video and data storage .
- Add version control for recorded meetings and summaries , so users can easily find and refer to older sessions when needed .

3. **Mobile Compatibility**

- Create a fully optimised mobile version that lets users join and participate from anywhere without losing any major functionality .

4. **Customisation and User Personalisation**

- Allow users to change themes , pick personal avatars , and set workspace preferences to make the overall experience feel more engaging and flexible .
- Use AI to adjust workspace layouts automatically depending on meeting type or user activity for smoother usability and better comfort .

5. **Security and Scalability Improvements**

- Apply end to end encryption for every layer of communication to keep user data private and safe.
- Strengthen the backend infrastructure to handle more users at once and support larger enterprise level use.

These future improvements will help Collabio grow into a complete and intelligent virtual collaboration system that smoothly blends human interaction with AI driven efficiency.

References

[1] Gunkel, Simon & Prins, Martin & Stokking, Hans & Niamut, Omar. (2017). WebVR meets WebRTC: Towards 360-degree social VR experiences. 457-458. 10.1109/VR.2017.7892377.

[2] Jovanović, A., & Milosavljević, A. (2022). VORTEX Metaverse platform for Gamified Collaborative Learning. *Electronics*, *11*(3), 317. https://doi.org/10.3390/electronics11030317

[3] Petrykowski, M., Berger, P., Hennig, P., & Meinel, C. (2018). Digital Collaboration with a Whiteboard in Virtual Reality. In *Advances in intelligent systems and computing* (pp. 962–981). https://doi.org/10.1007/978-3-030-02686-8_72

[4] E. O Nuallain, M. Diop, and A. Wade, "Virtual classroom solution with WebRTC in a collaborative context in mathematics learning situation," in *Proc. Int. Conf. Adv. Intell. Syst. Comput.*, Jul. 2022, pp. 1–10. [Online]. Available: https://www.researchgate.net/publication/361774251.

[5] Doumanis, I., Economou, D., Sim, G. R., & Porter, S. (2018). The impact of multimodal collaborative virtual environments on learning: A gamified online debate. *Computers & Education*, *130*, 121–138. https://doi.org/10.1016/j.compedu.2018.09.017

[6] Mahmoud, H., & Abozariba, R. (2024). A systematic review on WebRTC for potential applications and challenges beyond audio video streaming. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-024-20448-9

[7] V. Vanduhe, M. F. Hassan, and D. M. Nat, "Gamified collaborative environment in Moodle," *Int. J. Electr. Comput. Eng.*, vol. 11, no. 5, pp. 4567–4575, Oct. 2021. [Online]. Available: https://www.academia.edu/53497476.

[8] Asthana, S., Hilleli, S., He, P., & Halfaker, A. (2025). Summaries, highlights, and action items: Design, implementation and evaluation of an LLM-powered Meeting Recap System. *Proceedings of the ACM on Human-Computer Interaction*, *9*(2), 1–29. https://doi.org/10.1145/3711074

[9] Tepper, N., Hashavit, A., Barnea, M., Ronen, I., & Leiba, L. (2018). Collabot. *IBM*. https://doi.org/10.1145/3159652.3160588

[10] "WebRTC Documentation," [Online]. Available: https://webrtc.org/.

[11] "LivekitCloud Documentation," [Online]. Available: https://docs.livekit.io/.

[12] "p5.js Reference," [Online]. Available: https://p5js.org/reference/.

[13] "Socket.io Documentation," [Online]. Available: https://socket.io/docs/.

[14] "Tldraw Documentation," [Online]. Available: https://tldraw.dev/.

Proforma 4

Undertaking from the UG/PG student(s) while submitting his/her final dissertation to his respective institute

Ref. No. _____

I / We, the following student(s)

Sr. No.	Sequence of students names on a dissertation	Students name	Name of the Institute & Place	Email & Mobile
1.	1st student	Manya Lamba	SIT, Pune	manya.lamba.btech2022@sitpune.edu.in 9575277917
2.	2nd student	Nakul Kushwaha	SIT, Pune	nakul.kushwaha.btech2022@sitpune.edu.in 8319951533

hereby give an undertaking that the dissertation entitled **Collabio: Collaboration Made Easy** has been checked for its Similarity Index/Plagiarism through **Turnitin** software tool; and that the document has been prepared by me/us and it is my/our original work and free of any plagiarism.

It was found that:

1.	The Similarity Index (SI) was: *(Note: SI range: 0 to 10%; if SI is >10%, then authors cannot communicate ms;* ***attachment of SI report is mandatory****)*	2 %
2	The Artificial Intelligence Index (AI) was	* %
2.	The ethical clearance for research work conducted obtained from: *(Note: Name the consent obtaining body; if 'not appliable' then write so)*	Not applicable
3.	The source of funding for research was: *(Note: Name the funding agency; or write 'self' if no funding source is involved)*	Self
4.	Conflict of interest: *(Note: Tick √ whichever is applicable)*	No
5.	The material (adopted text, tables, figures, graphs, etc.) as has been obtained from other sources, has been duly acknowledged in the manuscript: *(Note: Tick √ whichever is applicable)*	Yes

In case if any of the above-furnished information is found false at any point in time, then the University authorities can take action as deemed fit against all of us.

Manya Lamba Nakul Kushwaha Prof Ranjeet Bidwe

Full Name & Signature of the student(s)

Name & Signature of SIU Guide/Mentor

Date: 05-11-2025

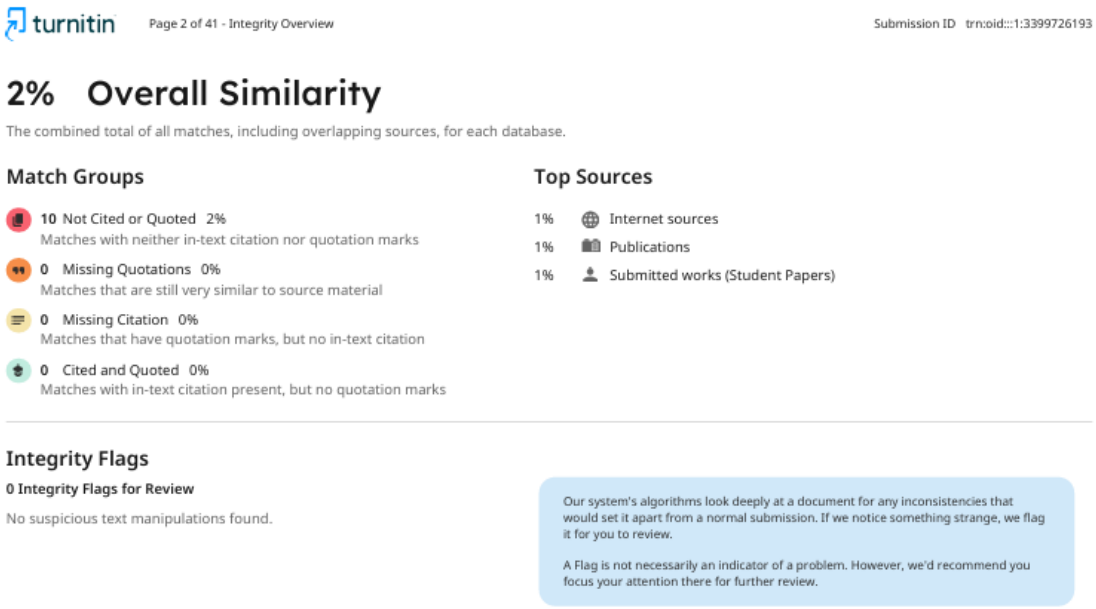
Endorsement by

Place: Pune Academic Integrity Committee (AIC)

Note: It is mandatory that the Similarity Index report of plagiarism (only first page) should be appended to the UG/PG dissertation

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Similarity Report



turnitin Page 2 of 41 - Integrity Overview Submission ID trn:oid:::1:3399726193

2% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **10** Not Cited or Quoted 2%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
1% Publications
1% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

AI Plag Report

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Patent Publication Document

Below document was submitted for the patenting of the above project

CROSS-REFERENCE TO RELATED AI AND PRIORITY

The present application claims no priority from any of the patent application(s).

FIELD OF THE INVENTION:

The invention pertains to web-based collaboration solutions of virtual meetings with an emphasis on natural spatial interactions and dynamic navigation in virtual spaces, in which conventional video call applications don’t support effective teamwork.

BACKROUND OF INVENTION

Normal video call apps such as Zoom and Google Meet are stiff and don't have the ability to connect in a natural manner like in-person meetings. Following are systems attempting to enhance cooperation.

Patent IN-DEL-2008-01273A outlines a web-based teamwork platform available on a variety of devices and operating systems. The platform allows multiple individuals to collaborate on common documents in real-time, and all changes appear immediately. The platform features chat capabilities through text, voice, and video, and also a virtual whiteboard with pencils, shape drawing tools, text boxes, and the ability to import images. Users may record sessions to look back on, save documents in many formats, and add custom plugins to add additional functionality.

Patent US2022124284A1 discloses a virtual conferencing system in which users are presented as avatars in a three-dimensional space. The system facilitates avatar movement in a virtual space with video and audio streams that adapt with avatar proximity to deliver natural interactions. The system uses sophisticated rendering of 3D graphics and includes spatial audio adapting with avatar positions. Users are also allowed to personalize their avatar look and set virtual camera angles to achieve different viewpoints.

Patent US2022070232A1 describes a virtual collaborative tool with avatars moving in virtual rooms in 2D or 3D using keyboard commands. Avatars initiating video calls automatically when in close proximity and stopping when moving away replicate in-person interactions. The tool includes public and private rooms with controllable permissions settings to permit users to define event spaces, share multimedia files, and publish notices to rooms.

Patent US2017024100A1 constructs a file and application sharing tool by dragging to a dedicated screen location. The tool is compatible with all devices and depends on eye tracking, voice instructions, or gestures to control it with ease. Work shared remains

accessible even when users are logged out, and text overlays assist in low-bandwidth environments.

The above systems referenced include some teamwork aspects but none of them integrate proximity-triggered video calls gamified movement with keys and touches and a chat for entire group. These systems lack in providing dynamic, natural meetings like real life interactions. For example, Patent IN-DEL-2008-01273A is centered on document editing and whiteboards with chat but does not utilize avatars or spatial triggers and hence becomes less dynamic. Patent US2022124284A1 includes avatars and proximity-triggered audio-video adaptations but don't initiate calls automatically and involve heavy 3D calculations and hence might limit accessibility on low-end devices. Patent US2022070232A1 includes spatial calls on avatar touching but is based on precise contact, does not have mobile users' touch controls, and lacks a universal chat for all. Also, such systems are usually software-specific or require complicated setups like additional plugins or high-end hardware to perform rendering in 3D. In contrast to it, proposed system employs a web browser exclusively to make it hardware-free and simple to use, integrating proximity-triggered video calls with dynamic movement and animation with keys and touches and a global chat to make teamwork active, dynamic, and easy with a synced whiteboard for creative collaboration.

OBJECTIVES OF THE INVENTION:

The main goal is to improve virtual meetings with a platform using simple web tools cutting down setup time and giving users a natural, lively experience.

Other goals include:

(i) To spark engagement with fun navigation and quick communication.

(ii) To help users connect by showing nearby avatars and group messaging.

(iii) To keep it open to all with no need for extra equipment.

SUMMARY OF THE INVENTION:

A robust collaboration platform today requires wise tools to capture and exchange data quickly and to share it fast and respond to it accordingly. Collabio is a fresh solution providing a simple, dynamic means to collaborate online. By delivering data from virtual 2D space to users and displaying real-time updates on web app it makes it easier to manage meetings in fast-paced virtual workplaces, where it is difficult to build active and efficient teamwork when efficiency matters.

Collabio employs sensing layer (WebRTC, Socket.io) to capture avatar movements and user inputs, communication layer (LivekitCloud) to manage data, and platform layer (React.js, p5.js) to render active graphics. Proximity video calls begin when avatars are closer than 100 pixels across 2D space, movement gamified avatar movement with joystick (nipplejs) on phone or keys (arrows and spacebar to jump) on a computer with moving avatars, global group conversation to talk simultaneously to all, synced whiteboard to draw together, and simple setup with just a web browser no hardware. The platform enables users to join meetings quickly, relocate to begin discussions, chat to entire group, and sketch concepts and ideas making virtual work natural and interactive.

DESCRIPTION OF THE DRAWINGS:

System Architecture (Figure 1): shows the system architecture of the Collabio platform. The user interacts using a web browser implemented on React and p5.js, responsible for handling avatar movement and UI. The request to join or start a meeting is given to the backend (Node.js using Express), which creates an access token to permit media communication securely. The token is used to connect to the LiveKit API and the Cloud to stream audio and video in real-time between users using peer-to-peer media streams. In parallel, real-time collaborative features like avatar movement, chat messages, and whiteboard changes are controlled using Socket.io. All events are synced to all users in real-time to give the virtual space a live and interactive experience. The whole system is hosted on the browser without needing any additional hardware and is thus as accessible and user-friendly as any normal web page.

Proximity Video Calling (Figure 2): shows the proximity-based video calling functionality in which users' audio and video are turned on automatically when their avatars are brought close to each other in virtual space. This mimics real-life interactions as it takes place in the virtual space. When 10 users are in a room divided into groups of 3, 3, and 4, each group can only see and video call members within their group. Once all 10 users come close together, they can all see and talk to each other via a shared video call.

User Experience (Figure 3): presents the user experience in Collabio from the landing page to leaving the virtual space. It demonstrates how the users proceed through the platform: watching the intro, adding in the meeting details, and joining a hosted or created room. Users are navigating as avatars when in the virtual space, participate in video calls, chat, and work on the whiteboard. User engagement aspects along the timeline are representative of the user’s feeling of ease of use at each step as well as interactivity and overall satisfaction with the flow with emoticons.

Avatar Movement (Figure 4): demonstrates how avatar movement is controlled by user input. Users are also allowed to move using both physical arrow keys and a virtual joystick (Nipple.js). The system computes direction and adjusts the avatar's location. Movement is synced by Socket.io and sent to all clients via the server. The avatar is rendered and animated on the p5.js canvas.

Video Logic (Figure 5): shows how Nearby Users Begin a Video Call, this illustration demonstrates how two avatars (Avatar A and Avatar B) initiate a video call when they are near to each other. Whenever they are close to each other and are less than 100 pixels away (step 503), a WebRTC video call begins automatically (step 504). Whenever they are away and are more than 100 pixels apart, the call automatically ends (step 505).

User Global Chatbox (Figure 6): depicts In-Game Global Chat System; this is a global chat feature whereby users are allowed to send and receive messages. In the photo, user "Manya" types "Hi" and another user responds with "Hey Manya.". The bottom box is used to type out messages visible to all the users in the room makes it easier to communicate when users are interacting in the virtual world.

Shared Whiteboard (Figure 7): shows a shared whiteboard where dummy users like Ram, Manya, and Nakul can be seen interacting and writing messages together in real time.

Setup Page (Figure 8): displays the Setup Page which enables users to fill out their name, meeting ID, and description previewing their video stream. It also offers to join a previously existing meeting or to start a new one. Users can also randomly spawn into one of a range of visually presented arenas shown below the form.

DETAILED DESCRIPTION OF THE INVENTION:

A few embodiments of the present invention, demonstrating all its features, may now be described in complete detail. The words "comprising," "having," "containing," and "including," and other forms, are utilized interchangeably and are open-ended to the extent that an item or items following any one or more of these words is not an exhaustive listing thereof or one restricted to only the mentioned item or items. It must also be appreciated that the singular determiners "a," "an," and "the" are to be construed as plural references except when the context specifically indicates otherwise. While any method which is similar or equivalent to those set out here may be utilized for practicing or testing embodiments of the present invention, the representative techniques are presented. The embodiments disclosed are only representative of the teaching of the present invention, which can be embodied differently.

It can be appreciated by and is hereby placed on notice by each and every reader of this written description that the disclosed example embodiments set out herein and claimed below can be practiced sufficiently well without recitation of any feature, element, or step, which is, or is not, specifically disclosed herein. For instance, use herein within this written description of phrases "one embodiment," "an embodiment," "an exemplary embodiment," and the like, means that the mentioned embodiment can include a certain feature, structure, or characteristic, but every embodiment may or may not include the certain feature, structure, or characteristic. The disclosed embodiments are simply representative of different forms or combinations. The use hereof is also not necessarily referring to the same embodiment. As well, when one particular feature, structure, or characteristic is

being discussed relative to an embodiment, it is submitted that it is within the knowledge of one skilled in the art to implement such feature, structure, or characteristic relative to other embodiments whether expressly discussed or not.

No terminology herein is to be construed to indicate any non-claimed component to be important or necessary. The inclusion herein of any and every one of the examples, or exemplary terminology (e.g., "such as") is only done to further clarify one or more example embodiments and doesn't narrow the scope of claims appended hereon except as otherwise set out.

The mentioning here of ranges of values is only done as a convenient shorthand reference to each individual discrete value within a range, and each discrete value is encompassed within the specification as if separately recited here. Where a disclosed range of values is expressed, it is to be interpreted that every intervening value, to the tenths of the unit of the lower limit unless context clearly directs otherwise, between the upper and the lower limit of the disclosed range and every other expressed or intervening value within the expressed range, is included within. All lesser ranges are included as well. The upper and the lower limits of the lesser ranges are included similarly, except for any excluded limit within the expressed range.

Collabio is a web-browser virtual collaboration platform to emulate natural real-life interactions in virtual meetings. In contrast to traditional video conferencing solutions with fixed grid layouts and constrained communication flows, Collabio presents a 2D virtual space with participants as avatars and interactions by proximity-triggered video and audio media, global chat, and whiteboards. The suggested platform has a low footprint, hardware-neutrality, and optimized for impromptu and interactive collaboration with minimal resource consumption.

The system functions solely by means of a common web browser and has no extra installs or hardware configuration to require, making it open to users from all walks and hardware capabilities to participate on the same footing. The system is accessed through a guided process starting with a landing page and a setup page to enter a meeting ID, name, and description and preview audio-video streams. Users thereafter proceed to enter the virtual world in which they are represented by avatars that are freely mobile in a 2D plane.

Every avatar can be moved using the arrow keys on desktop or through a small virtual joystick made with Nipple.js on mobile. The movement feels smooth and kind of natural across devices. When two avatars come close , around 100 pixels apart , the system automatically starts a peer to peer video and voice call using WebRTC. Once they move away from each other , the call ends by itself , helping to save bandwidth and avoid unnecessary distractions.

The system is divided into three layers , the sensing layer , communication layer and the platform layer. The sensing layer uses WebRTC and Socket.io to send and receive data about avatar movement and user input in real time. The communication layer depends on LiveKit Cloud to handle audio and video stability while keeping the delay really low. The platform layer is built with React.js and p5.js and it takes care of drawing avatars and interface elements inside a live canvas space.

A global chat option lets users talk freely across rooms without needing to be close. The whiteboard built with Tldraw helps people draw , share ideas and brainstorm together in real time. Every change made on the whiteboard gets updated instantly for all users using Socket.io , making teamwork easy even when everyone is remote.

The backend is made with Node.js and Express. It takes care of APIs , creating meeting rooms , and checking access tokens through LiveKit SDK. The backend also controls live updates for avatar movement , whiteboard sync and global chat messages. It keeps everything efficient by closing inactive sessions and removing socket connections when users leave.

The figures in the report show how everything works together. For example , Figure 2 shows how video calls start when avatars come near each other , Figure 4 explains avatar movement , Figure 6 and 7 show the chatbox and shared whiteboard , Figure 5 covers how video sessions change with distance , and Figure 3 shows how users move from the landing page to the virtual room.

The interface is styled with Tailwind CSS to keep it simple , neat and responsive. The focus is on real time engagement , with random background themes and clean avatar looks that mix a bit of fun with a professional feel.

Critical security and performance issues are addressed by bandwidth optimizations through idle stream termination, sequence-controlled avatar movement to eliminate lags or update conflicts, and debounced whiteboard synchronizations to avoid spamming. Scalability is also ensured through the architecture to cater to multiple users in a single virtual room without performance constraints.

Collabio's innovative aspects solve issues existing in the prior art. For instance, whereas other systems like Zoom or Google Meet necessitate breakout sessions by hand, natural group separation is accomplished by avatar proximity on Collabio. Solutions like Gather.Town are built around cartoonish interfaces and are typically cramped, and systems like FrameVR are costly hardware-intensive solutions. Collabio avoids all of these by providing a low-latency, browser-native experience mimicking real movement, unstructured conversation and collaborative creativity.

THRESHOLD PARAMETERS

Proximity Range	100px
Number of Users per Room	5
Joystick Sensitivity:	A full joystick push of 50 px makes the avatar move 100 pixels per second.
Video Resolution	In typical operation, the video resolution is approximately 480p (640×480 pixels), depending on network conditions and device capabilities.

TECHNOLOGY STACK WITH VERSIONS

Below mentioned are the technologies that have been used in the development of collabio

Frontend		
UI/UX	**Tailwind CSS**	^3.4.17
Canvas	**P5**	^1.11.3
Whiteboard	**Tldraw**	^3.10.3
Video Transmission	**Live-kit client**	^2.9.8
Website development	**React.js**	^18.3.1
Backend		
API endpoints	**Node.js/express**	^4.21.2
Video Transmission	**Live Kit-server-sdk**	^2.10.2
Message Transmission	**Socket.IO**	^4.8.1
Language	**TypeScript**	^5.2.2

EXPERIMENTAL RESULTS AND TESTING

Use Case	**Observed Result**	**Description**

Proximity Video Call	Connects in under 1 second	Users' avatars come close → WebRTC starts video instantly using LiveKit Cloud TURN relay.
Group Merge Video Latency	Delay is < 1 second	When multiple users join, ICE negotiation stays fast via LiveKit's TURN, no visible call drop.
Low Bandwidth Handling	VP8 auto-drops to 180p resolution	Uses simulcast streams; adjusts video size on weak networks without stopping the call.
Global Chat & Movement	Real-time with no visible lag	Socket.io transmits movement/chat instantly; feels like walking/talking in real life.
Whiteboard Synchronization	Instant updates, zero lag	Users draw/erase together; powered by real-time WebSocket events, no delay seen.
Random Avatars/Backgrounds	Rendered instantly with good internet	Switching avatars & backgrounds does not cause frame drops or performance drops.

AUDIO AND PERFORMANCE METRICS

Metric	**Measured Value**	**Explanation**
Audio Codec	Opus (48kHz Stereo)	High clarity voice; minptime=10, usedtx=1 for lower idle bandwidth.
Video Codec	VP8, L1T3 mode	Smooth video encoding with resolution fallback & scalable streams.
Audio Jitter	~2.2 ms	Consistent audio quality, no noticeable jitter or stutter.

Video Jitter	~6 ms	Good for face-to-face; small enough not to affect lip sync or gestures.
Round Trip Time	44–48 ms	Very low latency for both voice & video; suitable for spontaneous chats.
Frames Dropped	0	Video runs smooth even during avatar jumps & camera switch.
Encode Time	~1.5 sec total	Shows efficient CPU/GPU handling; no hardware stress on Chrome.

BANDWIDTH, NETWORK AND LIVEKIT CLOUD

Aspect	**Value**	**Explanation**
Audio Bitrate	~48 kbps	Balanced for clear voice while saving bandwidth.
Video Bitrate	~121 kbps	Good video quality without overloading weak connections.
ICE State	connected	Peer-to-peer link stable; fallback via LiveKit TURN relay when direct is not possible.
TURN Server Used	livekit.cloud (Relay IP)	Media routed securely through LiveKit Cloud TURN → ensures stable connectivity globally.
Candidate Pair	relay (UDP)	TURN relay used for NAT/firewall traversal; works behind corporate networks too.
Available Outgoing Bitrate	~1.2 Mbps	Sufficient for multiple simultaneous users per room.
CPU/Memory Stress	None detected	Chrome never triggered CPU or quality limits during 20+ mins test runs.

OVERALL STRESS TEST RESULTS

Test	Result	Explanation
Group Size	Tested with **6–8 users**	Multiple video/audio streams handled with no frame drop or lag.
Whiteboard Delay	Near-zero sync time	Large drawings sync instantly for all.
Simulcast Fallback	Automatic resolution drop	Users with low network stay connected with video downgraded to 180p.
Security Layer	DTLS + TURN encryption	All streams encrypted end-to-end, LiveKit handles secure relay via DTLS-SRTP.

STATEMENT OF TECHNICAL ADVANCEMENT

Collabio is a web-based virtual collaboration platform that makes collaboration appear natural with proximity-based video calls, a shared whiteboard, and a global group chat all working together in a typical web browser. It needs no top-dollar GPU, VR gear, or special equipment a decent internet connection suffices.

When participants move their avatars close to each other in virtual space, WebRTC automatically starts up their voice and video. When they move away, both get disconnected. This keeps proceedings running smoothly while also saving bandwidth. The whiteboard enables people to sketch together in real time, while everyone is able to send messages to the whole room any time via the group chat.

As Collabio also runs purely from within a web browser, it is still lightweight, hardware-independent software that anyone can use on any standard device without further setup. This facilitates more interactive, engaging online meeting experiences for everyone that are simple to join.

Patent Communication

Patent Filing Communication Timeline – "Collabio : Collaboration Made Easy"

1. Initial Submission (May 4 , 2025)

Details of the invention titled "Collabio : Collaboration Made Easy" were submitted for patent filing.

The project was described as a gamified, interactive virtual collaboration platform enabling video calls, live chatting, and synchronized whiteboard sharing.

2. Prior Art Report & Draft Instructions (May 5 , 2025)

The prior–art search report and one sample patent draft were shared. Instructions were given to :

- Identify 4 – 5 most relevant patents from the report.
- Highlight major differences between those patents and the proposed invention.
- Mention the comparative analysis under "Background of the Invention."
- Prepare & share the draft (excluding claim part).

3. Draft Submission (May 19 , 2025)

A full draft of the patent specification (excluding claims) was submitted along with related technical drawings.

The "Background of the Invention" section included prior–art analysis and mentioned the key differences from the proposed system.

4. Request for Additional Technical Details (July 10 , 2025)

To complete the Indian patent specification, some extra technical details were requested :

- Reference numerals for all figures (e.g. user (101), interface (102), canvas (103), etc.)
- Threshold parameters – proximity range, maximum users per room, frame rate / video resolution , joystick sensitivity.
- Detailed software stack : ReactJS, Tailwind CSS , Node.js / Express versions , Livekit setup , WebRTC codecs.
- Experimental results : latency , memory use , frame drop rate , stress testing outcomes.
- Best mode of performance and deployment setup.

- Statement of technical advancement — highlighting Collabio's novelty and its lightweight browser–native design.

5. Draft Modifications (July 10 , 2025)

The requested changes were assigned to the student team for incorporation into the previous draft.

6. Document Re–submission (July 25 , 2025)

Revised documents were submitted after integrating all received feedback and technical adjustments.

7. Status Inquiry (Sept 4 , 2025)

A request was sent for an update regarding the application status after re–submission of the modified draft.

8. Follow up for Expedited Processing (Sept 18 , 2025)

A follow–up mail was shared requesting faster processing of the patent application since the applicants were final–year B.Tech students, and delay could impact evaluation timelines.

9. Updated Forms Submitted (Sept 22 , 2025)

Updated patent forms were sent as per received suggestions, with a note requesting verification and confirmation if any more edits were needed.

10. Final Confirmation Request (Oct 7 , 2025)

A final message was sent requesting confirmation about correctness of the latest form updates and whether any further revisions were still required.

Project Recognitions

1. We presented our project “Collabio: Collaboration Made Easy” at SymbiTech 2025 on *19th September 2025*, where it was shortlisted for display and demonstration. The project was showcased in front of Dr. Abhay Jere, Chief Innovation Officer of India.

2. We also had the opportunity to present our project to Ms. Lily Ley, Vice President & Chief Information Officer of PACCAR, who appreciated our innovative implementation and real–time collaboration features.