WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

ClassView (Secure LAN-Based Real-Time Screen Monitoring & Collaboration System)

Mr. Mayank Chauhan[1],Kumaran S[2],Lokeshwar SG[3],Kishore M[4],Hari Prasad K[5]

Department of Artificial Intelligence and Data Science
Sri Shakthi Institute of Engineering & Technology, Coimbatore.

ABSTRACT :

ClassView is a secure, LAN-based real-time screen monitoring and collaboration system designed to facilitate effective supervision in computer laboratories and educational environments. The system enables teachers to view students' computer screens through a centralized dashboard when students explicitly authorize screen sharing. It aims to reduce the limitations of manual monitoring and improve the efficiency of practical classroom activities.

The proposed system utilizes HTML, CSS, and JavaScript for the frontend, Python with Django for backend services, WebSockets for real-time signaling and communication, and WebRTC for transmitting live screen streams. A database maintains relevant user, classroom, and session information. Students can join authorized classroom sessions, initiate screen sharing through browser permissions, and stop sharing at any time. The teacher dashboard displays student connection statuses and provides access to individual live screen streams.

ClassView emphasizes real-time communication, privacy, user consent, and controlled classroom access. By integrating web technologies with peer-to-peer media communication, the system provides a practical approach to digital classroom supervision. The proposed system can be further enhanced with multiple classroom support, collaborative tools, and improved network reliability.

Keywords: ClassView, WebRTC, Django, WebSockets, Screen Sharing, LAN, Classroom Monitoring.

DIAGRAM

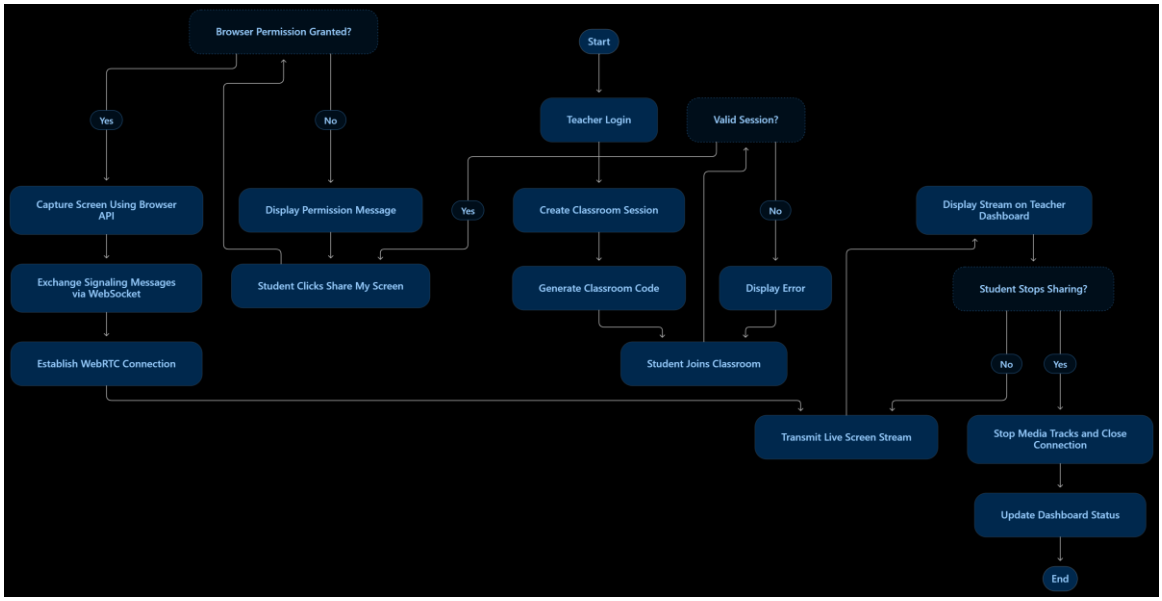


Figure 1. Flow Diagram of the ClassView System.

Introduction

The rapid development of information technology has transformed the way teaching and learning activities are conducted in educational institutions. Computer laboratories play an important role in practical learning, programming, and technical training. However, monitoring the activities of multiple

students simultaneously can be challenging for teachers, particularly when students work on different computers.

ClassView is a secure, LAN-based real-time screen monitoring and collaboration system designed to improve classroom supervision. It enables teachers to view students' computer screens through a centralized dashboard when students explicitly authorize screen sharing. The system allows students to join a classroom session using an authorized session code and share their screens through a web browser.

The system uses web technologies such as HTML, CSS, and JavaScript to develop the user interface, Python with Django for backend services, WebSockets for real-time signaling and status updates, and WebRTC for transmitting live screen streams. A database maintains relevant user, classroom, and session information. Devices connected to the same local area network can communicate through the application, subject to the network and browser security configuration.

The primary objective of ClassView is to simplify practical classroom supervision, reduce the effort involved in manually checking individual computers, and help teachers observe students' progress more efficiently. The system emphasizes user consent, authorized access, and transparent screen sharing. Students retain control over their screen-sharing sessions and can stop sharing at any time.

ClassView provides a foundation for future enhancements, including multiple classroom management, teacher feedback tools, collaboration features, and improved connection reliability. Thus, the proposed system aims to support a more efficient, interactive, and privacy-conscious digital learning environment.

Literature Review

The development of real-time screen monitoring and collaboration systems has gained importance with the increasing use of computers in educational institutions. Existing technologies such as remote desktop applications, web conferencing platforms, and browser-based screen-sharing tools provide different approaches to remote interaction and computer monitoring. A review of these technologies helps identify suitable methods for developing ClassView.

Remote Desktop and Computer Monitoring Systems

Remote desktop systems allow users to view and interact with another computer over a network. These systems are useful for technical support and remote administration. However, they may require additional software installation, configuration, or administrative privileges. ClassView focuses on browser-based screen sharing to simplify access for students and teachers.

Web-Based Screen-Sharing Technologies

Modern web browsers provide screen-capture functionality through the getDisplayMedia() API. This API enables users to select and share a screen, window, or browser tab after granting permission. It reduces the need for separate screen-capture software and supports a user-controlled sharing process. ClassView uses this capability to allow students to initiate and stop screen sharing explicitly.

WebRTC for Real-Time Communication

Web Real-Time Communication (WebRTC) is a technology that enables real-time audio, video, and data communication between supported browsers. It establishes peer connections using session negotiation and Interactive Connectivity Establishment (ICE). WebRTC is suitable for ClassView because it can transmit live screen streams with low latency under appropriate network conditions.

WebSockets for Real-Time Signaling

WebSockets provide a persistent, bidirectional communication channel between a client and a server. They are useful for exchanging classroom events, connection statuses, and WebRTC signaling messages, including offers, answers, and ICE candidates. In ClassView, WebSockets support coordination between teacher and student browsers, while WebRTC carries the actual screen-video stream.

Web Frameworks and Database Management

Web frameworks such as Django provide structured methods for implementing authentication, routing, session management, and database operations. Relational databases such as SQLite and PostgreSQL can store user details, classroom information, and session records. These technologies provide a foundation for developing and maintaining ClassView's backend services.

Research Gap and Motivation

Existing remote-access and screen-sharing technologies provide useful communication features, but a classroom-focused solution must combine authorized student participation, a centralized teacher dashboard, accurate connection statuses, and straightforward screen-sharing controls. ClassView addresses these requirements through an integrated web application that combines browser-based screen capture, WebRTC media transmission, WebSocket signaling, and classroom management.

Summary

The reviewed technologies provide the technical foundation for developing ClassView. Their integration can support real-time classroom screen sharing while maintaining user consent and controlled access. Further testing is required to evaluate connection reliability, performance with multiple students, and usability under different LAN conditions.

Methodology

The methodology of ClassView involves designing and developing a secure, LAN-based web application for real-time screen sharing between students and teachers. The system integrates frontend development, backend processing, browser-based screen capture, WebRTC communication, and database management to provide an efficient classroom monitoring solution.

The system follows a client-server architecture in which the teacher and students access the application through web browsers connected to the same local area network (LAN). The teacher logs in and creates a classroom session with a unique code. Students join the session using the authorized code, and the backend verifies their access before allowing participation.

After joining the classroom, students can select the *Share My Screen* option. The browser displays its native screen-selection and permission interface, and screen capture begins only after the student grants permission. Students can view their sharing status and stop sharing at any time, ensuring transparency and user control.

WebSockets are used to exchange signaling messages and real-time connection-status updates between the teacher and student browsers. WebRTC establishes the media connection through offer-answer negotiation and ICE candidate exchange. Once the connection is established, the student's selected screen is transmitted as a live video stream to the teacher's dashboard.

The teacher dashboard displays student identifiers, connection statuses, and live screen previews. Teachers can select an individual preview to view the shared screen in a larger display. The dashboard updates when students join, start sharing, stop sharing, or disconnect. A database stores relevant user, classroom, and session information to support session management and access control.

The system is tested incrementally, beginning with one teacher and one student using two laptops connected to the same LAN. Testing includes authentication, classroom joining, browser screen-capture permissions, WebSocket signaling, WebRTC connection establishment, live screen display, stopping screen sharing, and disconnection handling. Multiple-student testing is performed after the basic connection works reliably.

Security and privacy are maintained through authentication, classroom-level authorization, explicit browser permissions, and proper cleanup of media tracks and connections. Secure HTTPS/WSS configuration is used where required by browser security policies. The system does not perform hidden screen capture or bypass user permissions.

The overall methodology progresses from requirement analysis and system design to frontend and backend development, classroom management, screen capture, real-time communication, dashboard integration, and testing. This approach aims to produce a functional, reliable, and privacy-conscious screen-sharing system for educational environments.

Proposed Solution

The proposed system, ClassView: Secure LAN-Based Real-Time Screen Monitoring & Collaboration System, provides a secure and efficient platform for teachers to monitor student screens during classroom sessions or laboratory activities. The system operates over a local area network (LAN), allowing teachers and students connected to the same network to communicate without depending on external screen-monitoring services.

The system uses a web-based interface developed with HTML, CSS, and JavaScript, with Python and Django handling backend operations. Teachers can log in, create classroom sessions, and share a unique session code with students. Students can join the session and initiate screen sharing by selecting their screen or application window through the browser's built-in screen-sharing permission dialog.

WebRTC is used to transmit live screen streams, while WebSockets support signaling, session coordination, and real-time status updates. The teacher dashboard displays the screens of students who have authorized sharing, allowing teachers to observe ongoing activities and identify technical difficulties. A database stores essential information such as user accounts, classroom sessions, and sharing-session details.

Security and privacy are key aspects of the proposed solution. Screen sharing begins only after the student grants permission, and students can stop sharing at any time. Authentication, classroom access validation, secure communication, and proper session termination help prevent unauthorized access. The proposed system aims to improve classroom supervision, support practical laboratory sessions, and encourage interactive learning while respecting student privacy. Future enhancements may include file sharing, teacher-to-student communication, session reports, and support for larger classrooms.

Modules

User Authentication Module

This module provides secure login and authentication for teachers and students. It verifies user credentials and ensures that only authorized users can access the system.

Classroom Session Management Module

This module allows teachers to create classroom sessions and generate unique session codes. Students can use these codes to join the appropriate classroom session.

Student Screen Sharing Module

This module enables students to share their screens after granting permission through the browser's built-in screen-sharing dialog. Students can select a screen, application window, or browser tab and stop sharing whenever necessary.

Real-Time Communication Module

This module uses WebRTC to transmit live screen streams between student devices and the teacher dashboard. WebSockets handle signaling, connection setup, and real-time session status updates.

Teacher Dashboard Module

This module provides a centralized interface for teachers to view the screens of students who have authorized sharing. It displays student details, connection status, and live screen previews.

Database Management Module

This module stores and manages user details, classroom information, session records, and screen-sharing status. Django ORM with SQLite or PostgreSQL can be used for database operations.

Security and Privacy Module

This module protects the system through authentication, classroom access validation, secure communication, and proper session termination. It ensures that screen sharing occurs only with the student's permission and that students can stop sharing at any time.

System Workflow

The ClassView system follows a structured workflow to enable secure, real-time screen sharing between students and teachers over a local area network (LAN).

Step 1: User Login

The teacher logs into the ClassView system using valid credentials. Students access the system through their respective devices.

Step 2: Create Classroom Session

The teacher creates a classroom session, and the system generates a unique session code for students to join.

Step 3: Student Joins the Session

Students enter the session code to join the classroom. The system validates the code and grants access to the appropriate session.

Step 4: Request Screen Sharing

Students click the "Share My Screen" option. The browser displays a permission dialog that allows students to select the screen, application window, or browser tab they want to share.

Step 5: Establish Communication

After permission is granted, the system uses WebSockets to exchange WebRTC signaling information, including connection offers, answers, and ICE candidates.

Step 6: Transmit Live Screen

WebRTC establishes a connection and transmits the authorized screen stream from the student's device to the teacher's dashboard.

Step 7: Display Student Screens

The teacher dashboard displays the live screen previews of students who are sharing their screens. Teachers can select an individual preview for a closer view.

Step 8: Stop Screen Sharing

Students can stop screen sharing at any time. The system terminates the media stream, closes the relevant connection, and updates the teacher dashboard.

Step 9: End Classroom Session

The teacher ends the classroom session when monitoring is complete. The system closes active connections and updates the session status.

Overall Workflow:

Teacher Login → Create Classroom Session → Generate Session Code → Student Joins → Student Grants Screen-Sharing Permission → WebSocket Signaling → WebRTC Connection → Live Screen Display on Teacher Dashboard → Stop Sharing → End Session.

Implementation and Results

The ClassView system is implemented as a web-based application designed for secure, real-time screen sharing between teachers and students connected to the same local area network (LAN). The frontend is developed using HTML, CSS, and JavaScript to provide a simple and user-friendly interface. Python and Django are used for backend processing, authentication, and classroom session management. WebSockets support real-time signaling and status updates, while WebRTC enables live screen transmission. SQLite or PostgreSQL can be used to store user information, classroom details, and session records.

During implementation, the teacher dashboard is designed to create classroom sessions, generate session codes, and display the screens of students who have authorized sharing. Students join the classroom using the session code and initiate screen sharing through the browser's permission dialog. Once permission is granted, the system establishes a WebRTC connection and transmits the selected screen to the teacher dashboard. Students can stop sharing whenever necessary, and the dashboard updates the connection status accordingly.

The system is evaluated through functional testing on devices connected to the same LAN. Testing focuses on user authentication, classroom session creation, student joining, screen-sharing permission, real-time stream transmission, dashboard updates, and connection termination. Additional testing is required to evaluate performance when multiple students share their screens simultaneously.

The expected result is a functional platform that enables teachers to view authorized student screen streams in real time through a centralized dashboard. The system is intended to improve classroom supervision, support laboratory activities, and facilitate interactive learning. Its performance depends on network bandwidth, device capabilities, browser compatibility, and the number of simultaneous connections.

The implementation emphasizes security and privacy by requiring explicit student permission before screen sharing begins. Students retain control over their screen-sharing sessions and can stop sharing at any time. Overall, ClassView demonstrates the practical application of WebRTC, WebSockets, and web technologies in developing a LAN-based classroom monitoring and collaboration system.

Note: Actual performance results, screenshots, response times, and multi-student testing measurements should be added after testing the completed application. Expected outcomes should not be presented as experimentally verified results.

Functional Results

The ClassView system is designed to provide secure and reliable real-time screen sharing between teachers and students connected to the same local area network (LAN). The functional results are evaluated based on the successful operation of the system's main modules.

User Authentication: The system allows authorized teachers and students to access the application using their login credentials.

Classroom Session Management: Teachers can create classroom sessions and generate unique session codes. Students can join the appropriate sessions using valid codes.

Screen Sharing: Students can select and share their screens through the browser's built-in permission dialog. Screen sharing starts only after the student grants permission.
Real-Time Screen Transmission: WebRTC enables the transmission of student screen streams to the teacher dashboard, while WebSockets support connection signaling and session status updates.
Teacher Dashboard: The dashboard is designed to display live screen previews and the sharing status of connected students, helping teachers supervise classroom activities.
Session Control: Students can stop screen sharing at any time. The system updates the dashboard and terminates the corresponding media stream and connection.
Security and Privacy: Authentication and session validation help restrict access to authorized classroom participants. The system does not perform hidden screen capture and requires explicit permission for screen sharing.
Overall, the functional design of ClassView supports classroom supervision, laboratory monitoring, and collaborative learning through a centralized web-based platform. The successful operation of each function should be verified through practical testing, and actual results should be documented with screenshots and test observations.

CONCLUSION

ClassView: Secure LAN-Based Real-Time Screen Monitoring & Collaboration System provides a web-based solution for improving classroom supervision and supporting interactive learning. By integrating HTML, CSS, JavaScript, Python, Django, WebRTC, and WebSockets, the system enables teachers to view student screens in real time over a local area network. The teacher dashboard simplifies classroom session management and allows teachers to monitor student activities during practical and laboratory sessions.
The system emphasizes security and privacy by requiring students to grant permission before sharing their screens and allowing them to stop sharing at any time. This approach supports transparent and authorized monitoring. ClassView can be further enhanced with features such as file sharing, real-time messaging, session reports, and improved support for multiple students. Overall, the proposed system demonstrates the potential of modern web technologies to create a secure, efficient, and collaborative digital learning environment.

REFERENCES

1. Mozilla Developer Network (MDN), “Screen Capture API,” Mozilla. Available: https://developer.mozilla.org/en-US/docs/Web/API/Screen_Capture_API
2. Mozilla Developer Network (MDN), “WebRTC API,” Mozilla. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API
3. Django Software Foundation, “Django Documentation.” Available: https://docs.djangoproject.com/
4. Django Channels, “Channels Documentation.” Available: https://channels.readthedocs.io/
5. WebRTC Project, “WebRTC: Real-Time Communication for the Web.” Available: https://webrtc.org/
6. IETF, “WebSocket Protocol,” RFC 6455, 2011. Available: https://www.rfc-editor.org/rfc/rfc6455
7. W3C, “Screen Capture,” W3C Specification. Available: https://www.w3.org/TR/screen-capture/
8. W3C, “WebRTC: Real-Time Communication in Browsers.” Available: https://www.w3.org/TR/webrtc/
9. Mozilla Developer Network (MDN), “MediaDevices: getDisplayMedia() Method.” Available: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getDisplayMedia
10. SQLite, “SQLite Documentation.” Available: https://www.sqlite.org/docs.html