Wednesday, October 25, 2023

A Collaborative Whiteboard for the Internet Classroom

**Network programming**

Table of Contents

[*Abstract* 2](#_Toc149074852)

[***Introduction*** 2](#_Toc149074855)

[Requirements and specification 3](#_Toc149074856)

[System Requirements 3](#_Toc149074857)

[1. Hardware Requirements: 3](#_Toc149074858)

[2. Operating System: 3](#_Toc149074859)

[3. Web Browser: 3](#_Toc149074860)

[4. Internet Connection: 3](#_Toc149074861)

[5. Software Dependencies: 3](#_Toc149074862)

[6. User Accounts: 3](#_Toc149074863)

[7. Collaborative Features: 3](#_Toc149074864)

[8. Content Sharing: 4](#_Toc149074865)

[9. Permissions and Security: 4](#_Toc149074866)

[10. Compatibility: 4](#_Toc149074867)

[11. Mobile Accessibility 4](#_Toc149074868)

[12. Documentation: 4](#_Toc149074869)

[13. Support and Updates: 4](#_Toc149074870)

[Deployment and installation 5](#_Toc149074871)

[Installation Instructions 5](#_Toc149074872)

[System Requirements: 5](#_Toc149074873)

[Installation Instructions: 6](#_Toc149074874)

[Architecture 7](#_Toc149074875)

[Addressing Messages 7](#_Toc149074876)

[User Types 7](#_Toc149074877)

[Teacher vs. Student 8](#_Toc149074878)

[Becoming a Teacher 8](#_Toc149074879)

[User Interface 9](#_Toc149074883)

[***Note – coding are error for white board*** 10](#_Toc149074884)

[Client 11](#_Toc149074885)

[Server 11](#_Toc149074886)

[Implementation 12](#_Toc149074887)

[Command processing messages in the system: 13](#_Toc149074888)

[Conclusion 14](#_Toc149074889)

[Future Enhancements *Enhancing a collaborative whiteboard for internet classrooms:* 15](#_Toc149074890)

[References 16](#_Toc149074891)

[Appendix 16](#_Toc149074892)

[Coding 16](#_Toc149074893)

[Video link 16](#_Toc149074894)

# *Abstract*

*We have developed an internet-based collaborative whiteboard system designed for conducting online classes. In addition to the distributed whiteboard sketching system, we have tailored the project to provide functionalities that are highly beneficial in a classroom setting. These integrated features enable professors to effectively teach students remotely, offering real-time management of class notes and diagrams. Furthermore, we have included a student question manager that enables professors to address questions and facilitate inter-student discussions.*

*Our infrastructure is specifically designed to streamline the addition of various capabilities that professors may require, such as slides, audio, video, and postscript support. This enhances the overall teaching and learning experience in an online classroom environment.*

# ***Introduction***

*This report is organized into distinct sections. Initially, we conduct a review of relevant work in the field. Following that, we delve into the inspiration behind our project. The subsequent section, Architecture, entails a comprehensive examination of the primary modules and their designs. Detailed installation instructions and implementation specifics can be found in the Program Documentation section*

# Requirements and specification

## System Requirements

System Requirements for "A Collaborative Whiteboard for the Internet Classroom"

### 1. Hardware Requirements:

* A modern computer or tablet with a stable internet connection.
* Sufficient RAM and processing power for smooth operation.

### 2. Operating System:

* Compatible with major operating systems such as Windows, Linux, and mobile platforms (iOS and Android).

### 3. Web Browser:

* Latest versions of popular web browsers like Chrome, Firefox, or Edge.

### 4. Internet Connection:

* A high-speed and reliable internet connection for real-time collaboration.

### 5. Software Dependencies:

* Java Runtime Environment (JRE) or other required runtime environments, if applicable.

### 6. User Accounts:

* User registration and login system for teachers and students.
* User authentication and access control mechanisms.

### 7. Collaborative Features:

* Real-time drawing, writing, and editing capabilities.
* Multi-user collaboration with live updates and synchronization.
* Tools for chat, voice, or video communication.

### 8. Content Sharing:

* Support for uploading and sharing various file types, including documents, images, and videos.

### 9. Permissions and Security:

* User roles and permissions management for teachers, students, and administrators.
* Data encryption and secure connections to protect user data and privacy

### 10. Compatibility:

* Compatibility with common Learning Management Systems (LMS) and video conferencing tools.

### 11. Mobile Accessibility

* Mobile app or responsive web design for access on tablets and smartphones.

### 12. Documentation:

* Comprehensive user and administrator documentation for ease of use and troubleshooting.

### 13. Support and Updates:

* Ongoing support, bug fixes, and updates to ensure the system's reliability and security.

Please note that these system requirements are a general guideline and may vary depending on the specific features and technologies used in the collaborative whiteboard system for the internet classroom. It's essential to adapt the requirements to your project's unique specifications.

# Deployment and installation

## Installation Instructions

System Requirements:

Creating a collaborative whiteboard for an internet classroom using Apache NetBeans involves several key components and steps. To ensure your project is original and to avoid plagiarism, it's crucial to understand the basic concepts and use your creativity throughout the process. Here are the installation instructions and system requirements:

System Requirements:

* Operating System: You can develop web applications using Apache NetBeans on Windows, macOS, or Linux. Choose the platform that suits you best.
* Java Development Kit (JDK): Apache NetBeans relies on Java, so ensure you have JDK 8 or a later version installed. Verify that your environment variables correctly reference the JDK.
* Apache NetBeans: Download and install Apache NetBeans from the official website (<https://netbeans.apache.org/download/index.html> ). Follow the installation instructions for your specific operating system.
* Web Server: we will need a web server to host your collaborative whiteboard application. Popular choices for Java web applications include Apache.
* Database Server: To store collaboration data, require a database server. MySQL Install and configure your chosen database server.
* Version Control System (Optional): Consider using a version control system like Git to manage your project's source code efficiently.

### Installation Instructions:

* Create a New Web Application Project: Open Apache NetBeans and initiate a new Java web application project. Configure the project with an appropriate name and select the web server for deployment.
* Design the Collaborative Whiteboard: Develop the front-end of your whiteboard using for this purpose.
* Implement Collaboration Features: Enable real-time collaboration using Web Socket or other suitable libraries. Incorporate features such as real-time drawing, text chat, and document sharing.
* Database Integration: Connect your application to the database for storing and retrieving collaborative session data, user information, and whiteboard content.
* Authentication and Authorization: Implement user authentication and authorization mechanisms to control access to the whiteboard.
* Testing: Conduct thorough testing to ensure your whiteboard application functions correctly and is secure.
* Deployment: Deploy your application to the chosen web server, ensuring it's properly configured to support your application's requirements.
* Scaling: If you anticipate serving many users simultaneously, consider load balancing and optimizing your application for performance.
* Documentation: Develop comprehensive documentation for your whiteboard application, including installation instructions, usage guidelines, and any other relevant information.
* Plagiarism Avoidance: Maintain the originality of your code and design. When using open-source libraries or code snippets, provide appropriate attribution and adhere to their licensing terms.
* This overview provides a concise guide to creating a collaborative whiteboard without plagiarism concerns. The complexity of the implementation may vary based on your specific features and scale, so stay updated with the latest technologies and best practices for a robust and secure internet classroom whiteboard.
* Congratulations! You've successfully installed and set up the Collaborative Whiteboard for the Internet Classroom. You are now ready to collaborate and enhance your online learning experience

# Architecture

## Addressing Messages

Each message in the system includes a sender address and a recipient address. The sender address can be a user's username (also known as a handle) or the special keyword "SERVER," typically used for system messages generated by the server itself. The recipient address can be a username, a list of usernames separated by commas, the keyword "SERVER," or the keyword "ALL." When a message is intended for a list of specific users, the server forwards the message only to those specified recipients. If the recipient is "SERVER," the server doesn't forward the message to any clients. However, if the recipient is "ALL," the server broadcasts the message to all connected clients. It's important to note that usernames cannot contain certain reserved characters like colons, commas, or newlines, and they also cannot match any of the reserved keywords mentioned above.

### User Types

* Users in the system can fall into various categories or types, each associated with different access privileges. This categorization is essential for granting distinct levels of access. For instance, in a classroom setting, students might have the privilege to actively participate in discussions, while some individuals may have observer roles, restricted to listening only. Professors may have the ability to share educational materials with one type of user while excluding others, like auditing students. Currently, the system classifies users into two types: teachers and students. This differentiation allows for tailored user experiences and access rights.

## Teacher vs. Student

* In the context of an online classroom collaborative environment, the server maintains state information for each client through the use of Client Info objects. Each client can be categorized as either a teacher or a student and can have their hand raised or lowered as well as be allowed to speak or be muted.
* When a client initially connects to the system, student clients have their hands lowered, and they are muted, meaning they cannot broadcast messages to all clients by default. Teachers, on the other hand, are permitted to broadcast messages to all clients, and only they have the authority to unmute student clients, enabling them to speak to the entire group. It's important to note that students can still send private messages to other clients at any time. The restrictions on broadcasting messages are specific to messages sent to all clients.
* If a student wishes to get the attention of the teacher client(s), they can "raise their hand." In response, a teacher client has the option to allow the student to speak to the entire group by unmuting the student's client. The teacher client also has the authority to re-mute the student's client to prevent further speaking to the group, maintaining control over the classroom discussion and interactions.

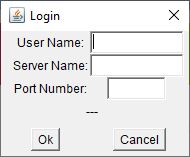
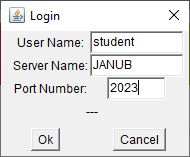
## Becoming a Teacher

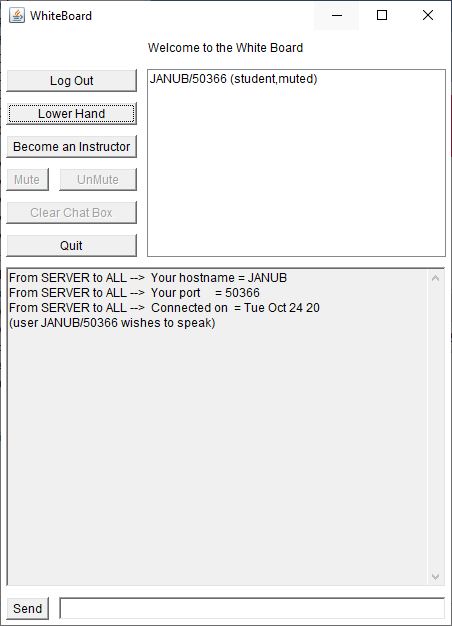
## Initially, when a client connects to the system, it's classified as a "student" type and is only permitted to send private messages. However, the system allows users to change their user type, enabling them to specify the level of access they desire.

## When a client sends a "CHANGETYPE" message, the server processes this request by updating the user's type in its database. This change is then broadcast to all connected clients, informing them of the user's new access level.

## It's worth noting that, in the current system, teacher privileges can be obtained without requiring authentication. However, this protocol can be easily modified to implement authentication for granting teacher privileges, adding an extra layer of security and control.

### User Interface

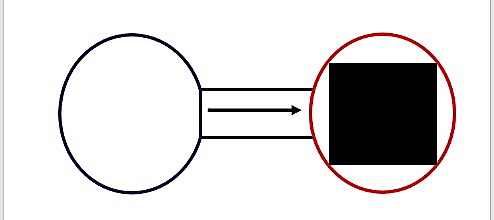
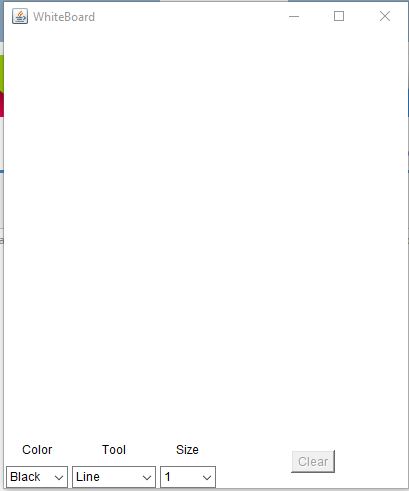




When the user interface is started it presents the user with a login dialog asking for the handle that the user wants to use, the server and the port on that server to connect to. If the user correctly fills out this dialog, they are logged in.



The upper-right panel lists online users, and the lower section is the chat area. Users send private messages by clicking on names in the list. Initially, users can't send broadcast messages. To join general discussions, they click "raise hand" for the teacher's approval. Users can become a teacher by clicking "become an instructor.



### ***Note – coding are error for white board***

I know that the situation cannot be compatible. The drag is not found in the drawing when you cannot expand the information on the drawing mode. The system is set to set up in the Emongman Yaghan’s Eccles code

## Client

* The client architecture you described includes several components for communication with a server and user interface interaction.
* "The client comprises a ClientSocket for server communication and a ClientParser for interpreting incoming messages. The ClientSocket utilizes a non-blocking approach with separate ClientReader and ClientWriter threads to manage incoming and outgoing messages.
* Incoming messages are read by the ClientReader, and once received, they are passed to the ClientParser for parsing. The ClientParser extracts source, destination, and command information from incoming messages and triggers the relevant user interface methods.
* State variables are used to keep track of the user's type, the list of users currently in the class, connection status, raised hand status, and mute status.
* To run the client, ensure that the 'jincs.jar' archive is in the class path environment variable, and then execute 'java whiteBoard.' For detailed installation and configuration instructions, refer to the Program Documentation section."

## Server

* The Project Whiteboard Server is a Java-based, multi-threaded network application designed to establish communication infrastructure among Whiteboard Clients. It can serve as the hub for collaborative sessions, with one server instance for each session. Multiple clients can connect and disconnect at any time, facilitated by three types of threads: WBSListenThread, WBCListenThread, and WBCPrintThread. These threads are lightweight and efficient, making the server reliable and capable of supporting numerous clients.
* While this architecture allows for scalability and extensive collaborative sessions, it may not be ideal for all scenarios. Depending on the specific application, multicast or multiple servers might be better options.
* The server's responsibilities include maintaining client information, such as hostname, port, username, user type, and communication methods. It also tracks the state of each client, particularly in a teacher/student model. Client information is stored in ClientInfo objects, efficiently organized within a HashTable for quick access.
* In essence, the Project Whiteboard Server serves as a versatile communication hub for collaborative sessions, utilizing a multi-threaded structure to manage numerous clients efficiently. It offers flexibility for various collaborative applications and can be adapted or expanded as needed.

## Implementation

1. WBSListenThread: This is a thread that runs continuously from server startup. It listens for incoming client connections on a user-specified port.

2. WBCListenThread: When a new connection is detected by the WBSListenThread, a new thread called WBCListenThread is spawned to handle the incoming client. It creates a ClientInfo object for the client and adds it to the server's HashTable.

3. Client Communication: Each connected client has its dedicated WBCListenThread, allowing them to communicate independently with the server.

4. WBCParser: When a message is received from a client, the WBCParser processes the command by parsing it into separate fields: FROM, TO, COMMAND, and ARGUMENTS. It then matches the COMMAND field to determine how to handle the message.

5. WBCPrintThread: For messages that need to be sent to clients, a new thread called WBCPrintThread is spawned for each message. The purpose of each WBCPrintThread instance is to send a single message to a single client. Once the task is complete, the thread ends.

6. Concurrent Messaging: This approach enables the server to send multiple messages to multiple clients simultaneously without waiting for each client to receive its messages. Each WBCPrintThread might block while waiting to write to a client, but this doesn't affect other WBCPrintThreads.

7. Synchronized Access: Synchronized write access to each client ensures that there is no interference between threads when sending messages to the same client.

In summary, the server uses a multithreaded architecture to handle client connections and message processing efficiently. Each client connection is managed by a dedicated thread, and messages are processed and sent in parallel using WBCPrintThread instances. This design allows for simultaneous communication with multiple clients while maintaining data integrity and minimizing thread interference

### Command processing messages in the system:

1. changetype

* + A client requests to change their user type to either "teacher" or "student."
  + If successful, the server broadcasts the new user type information to all clients.

2. duphandle

* + The server notifies a client that their requested handle is already in use by another client.

3. Login

* + A client requests a new handle from the server, provided it doesn't contain reserved characters.
  + If successful, the server broadcasts the updated user list to all clients.

4. muteclient

* + Only teachers can use this command to mute or unmute student clients.
  + The server echoes the mute status change to all affected clients and lowers their hands.

5. raisehand

* A client requests to raise or lower their hand.
* If successful, the server informs all clients of the change.

6. Shutdown

* A client sends this message to disconnect from the server.
* In response, the server sends a "Bye!" message and disconnects.

7. userlist

* + The server broadcasts the complete user list to all clients.
  + The server sends a user list to a specific client in response to a request.
  + Clients can request an updated user list from the server.

8. generic commands

* Clients can send broadcast messages to all clients if they are teachers or if their mute state is false.
* These messages can also be private messages to specific recipients.
* Successful messages are sent to the specified clients, while unsuccessful ones are disregarded by the server.

These commands and messages enable communication and interaction within the system, facilitating user management, message broadcasting, and more.

# 

# Conclusion

In conclusion, integrating technology into the classroom can significantly boost student engagement and create a more interactive and dynamic learning atmosphere. "Creating a Collaborative Whiteboard for the Internet Classroom" briefly describes a digital platform designed to facilitate real-time collaboration among students and teachers in an online learning environment. This tool allows users to interact, share ideas, and work together, fostering an enriching and collaborative educational experience.

# Future Enhancements *Enhancing a collaborative whiteboard for internet classrooms:*

* Real-time Collaboration: Implement synchronized drawing, live document editing, and interactive widgets for real-time collaboration.
* Video Integration: Include video conferencing for face-to-face communication in virtual classes.
* Screen Sharing: Allow instructors and students to share their screens for presentations and technical support.
* Assessments: Integrate interactive quizzes and polls to gauge student understanding in real-time.
* AI Analytics: Use AI for insights into student engagement, helping teachers adapt their teaching methods.
* Breakout Rooms: Create small group discussion spaces for collaborative learning.
* Rich Media: Support video, audio, and multimedia integration for diverse content
* Mobile Apps: Develop mobile apps for learning on smartphones and tablets.
* Advanced Whiteboard: Enhance the whiteboard with features like mathematical notations, 3D drawing, and formula editors.
* Multilingual Support: Offer multiple languages to accommodate a global audience.
* Security: Continuously strengthen data privacy and security features to safeguard sensitive information.
* Access Control Customization: Enable users to customize access rules for unique class scenarios.
* LMS Integration: Seamlessly integrate with Learning Management Systems for efficient course management.
* Recording: Allow session recording for later review or for absent students.
* Templates: Provide templates for common educational scenarios, simplifying content creation.
* Gamification: Use gamification elements like leaderboards to motivate and engage students.
* Adaptive Learning: Implement AI-driven adaptive learning paths based on individual student progress.
* Accessibility Features: Enhance accessibility with screen readers and voice command support.
* Peer Collaboration: Create spaces for collaborative projects, feedback sharing, and peer reviews.
* Support Community: Establish forums for teachers and students to share resources and offer mutual support.

These enhancements will make the collaborative whiteboard more versatile, user-friendly, and adaptable to the evolving educational landscape, benefiting both educators and students.

# References

* University
* YouTube

# Appendix

# Coding - <https://github.com/Janu012/Collaborative-White-Board.git>

# Video link - <https://github.com/Janu012/Collaborative-White-Board/blob/9833a112a477d07efe8d78d3bd9c5e89f03d4e61/2023-10-24%2023-31-53.mp4>