**A white rectangular frame with blue border

Description automatically generated**

**MINISTRY OF EDUCATION AND TRAINING**

**HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FAUCULTY OF INTERNATIONAL EDUCATION**

**🕯✡🕮🕮✡🕯**

**A logo with hands holding a book and a flame

Description automatically generated**

**Ho Chi Minh City, November 2023**

|  |  |
| --- | --- |
| ***Group of students involved:*** | |
| Nguyen Khac Huy | 21110773 |
| Le Bui Huu Phuc | 21110073 |

**Instructor: Nguyen Dang Quang**

**FINAL PROJECT REPORT:**

**IT PROJECT**

# Acknowledgment

We would like to express my sincere gratitude to Dr. Nguyen Dang Quang, our esteemed instructor, whose guidance and expertise have played a pivotal role in shaping our understanding and skills in the field of web application development.

Dr. Nguyen Dang Quang's dedication to imparting knowledge, fostering a collaborative learning environment, and providing invaluable insights into the intricacies of web application design have significantly contributed to the success of our collaborative whiteboard project.

His unwavering support, encouragement, and commitment to our academic and professional growth have been instrumental throughout the duration of this course. We are truly grateful for the opportunity to learn from such an accomplished and inspiring instructor.

Sincerely thanks.

# Preface

This report marks the culmination of our training journey, where our primary goal was to gain practical skills in introducing a web application. Within the limited timeframe, we not only developed the confidence to present our collaborative whiteboard app but also acquired fundamental IT knowledge essential for thriving in today's competitive landscape.

This effort represents a partial fulfillment of the course requirements. The report systematically organizes and presents the extensive program content, with each topic given its due prominence in individual chapters. By employing citation methods, we ensure the integrity of our references and recognize the insights shared by experts in web application development.

Our collective hope is that this report, with its structured chapters and content, proves to be a valuable resource for readers, particularly those interested in collaborative whiteboard web applications. Through this succinct preface, we express gratitude to our instructor and the broader community, anticipating that our work will contribute to the collective knowledge in the field.

Catalog

[Acknowledgment 3](#_Toc131790110)

[Preface 4](#_Toc940738713)

[I. Project description 5](#_Toc1220940177)

*[1.](#_Toc1135307754)* [Objectives 5](#_Toc1135307754)

*[2.](#_Toc725217883)* [Technologies Stack : 6](#_Toc725217883)

*[3.](#_Toc1767262856)* [User Benefits 7](#_Toc1767262856)

*[4.](#_Toc540499135)* [Use case diagram 8](#_Toc540499135)

*[5.](#_Toc313135135)* [Use case description tables 8](#_Toc313135135)

[II. Task Assignment 11](#_Toc1527278795)

[III. Design 13](#_Toc102674974)

*[1.](#_Toc1228919477)* [Process description 13](#_Toc1228919477)

*[2.](#_Toc2099416740)* [Class Design 15](#_Toc2099416740)

*[3.](#_Toc795292062)* [Graphic User Interface 19](#_Toc795292062)

[IV. Conclusion 20](#_Toc535467106)

*[1.](#_Toc1639169612)* [Limitations: 20](#_Toc1639169612)

*[2.](#_Toc1603445168)* [Difficulties 20](#_Toc1603445168)

*[3.](#_Toc330652373)* [Development ideas 20](#_Toc330652373)

[References 21](#_Toc1734238222)

# Project description

## Objectives

Collaborative board web applications are becoming increasingly popular in today’s world, where remote work is becoming more common. These applications allow users to collaborate on a virtual whiteboard in real-time, making it easier to share ideas and work together on projects.

In this report, we will be discussing the design of a collaborative board web application written on ReactJS and NodeJS. We will start by outlining the requirements of the program, including the data that will be used. We will then move on to designing the objects that will be used in the program, such as the objects on the canvas and how they will be managed to serve the purpose of saving and loading the canvas. We will also consider how the application server will connect with the client, and what technology we plan to use to achieve this. Additionally, we will consider how the client will transmit drawing data to the server, and how the server will broadcast this data to other clients.

After outlining the requirements and designing the objects, we will choose the front-end and back-end technologies that we will use to develop the application. We will then begin developing the application itself.

Finally, we will consider any challenges that we may face during the development process and how we plan to overcome them.

## Technologies Stack :

### *ReactJS :*

* Reusable Components: React enables the creation of reusable UI components, enhancing code maintainability and ensuring consistency across the project.
* Efficiency: Utilizes a virtual DOM, updating only the components that have changed in the real DOM, resulting in a more efficient and faster updating process.
* Real-time Updates: Ideal for real-time collaboration as data is transmitted to the server with each element update.

### *NodeJS:*

* Speed: Known for its speed, Node.js ensures a quick response time for web applications, contributing to overall application quality.
* Single Code Base: Developers write JavaScript for both client and server, simplifying data transfer and synchronization between them.
* Real-time Capabilities: Excellent for building scalable and high-throughput real-time web applications.
* Data Streaming: Provides faster data streaming without the need for buffering, enhancing overall performance.

### *Vite:*

* Fast Development Environment: Offers a faster and leaner development environment with instant server start-up and real-time updates reflected in the browser without full page reloads.
* Optimized Builds: Transforms source code into directly runnable formats, avoiding unnecessary bundling steps and optimizing build processes.
* Rich Features: Out-of-the-box support for TypeScript, JSX, CSS pre-processors, and more, enhancing development capabilities.

### *Overall Impact*

* Robust and Efficient Environment: The combination of ReactJS for dynamic UI, NodeJS for server-side efficiency and real-time capabilities, and Vite for a fast and optimized development environment creates a robust and efficient foundation for building a real-time collaborative board website.
* Project-Specific Considerations: The choice of technologies depends on the specific requirements of the project, ensuring that the selected stack aligns with the goals and objectives of the collaborative board web app.

## User Benefits

Similar to other drawing applications, Collaborative Board empowers users with a range of features, enabling them to draw, manipulate, and interact with graphic elements. The essential drawing functions include:

* Undo: The ability to revert to the previous action, providing a safety net for creative experimentation.
* Change Color: Users have the flexibility to alter colors using HSB, RGB, and Web formats, allowing for precise and customized color selections.
* Adjust Stroke Thickness: Users can modify the thickness of a stroke or adjust the border width of a shape, tailoring the visual appearance of their creations.
* Erase: The eraser function clears everything in its path, offering users control over their canvas. Additionally, users can adjust the size of the eraser for finer control.
* Pen: A versatile tool for free-form drawing strokes directly onto the canvas, facilitating artistic expression.
* Shape: Users can draw both rectangles and ellipses, enhancing the variety of elements that can be created on the artboard.
* Redo: Users can redo previously undone actions, allowing for greater flexibility and control in the creative process.

Beyond individual creativity, Collaborative Board distinguishes itself by offering internet connectivity, allowing users to invite one or more friends to collaborate on drawing projects. Once connected, users and their friends can engage in real-time chat, fostering communication and idea-sharing before collaboratively bringing their visions to life on the canvas. This collaborative aspect adds a social dimension to the drawing experience, making Collaborative Board a dynamic and interactive platform for creative expression.

## Use case diagram

A diagram of a diagram

Description automatically generated

*Image 1 – Use Case Diagram*

## Use case description tables

*Table 1 – Use case Draw Shape description*

|  |  |
| --- | --- |
| Use case name | Shape |
| Description | Allows host to draw a rectangle or a line on the board |
| Actor | Host |
| Preconditions | Click the Shape button and choose a specific shape |
| Conditions affecting termination outcome |  |

*Table 2 – Use case Pen description*

|  |  |
| --- | --- |
| Use case name | Pen |
| Description | Allows host to draw freely on the board |
| Actor | Host |
| Preconditions | Click Pen button |
| Conditions affecting termination outcome |  |

*Table 3 – Use case Eraser description*

|  |  |
| --- | --- |
| Use case name | Eraser |
| Description | Allows host to erase anything on the board |
| Actor | Host |
| Preconditions | Click Eraser button |
| Conditions affecting termination outcome | The server is running, the connection is established successfully The server is terminated, connection failed |

*Table 4 – Use case Undo/Redo description*

|  |  |
| --- | --- |
| Use case name | Undo/Redo |
| Description | Allows host to undo/redo the latest action on the board |
| Actor | Host |

|  |  |
| --- | --- |
| Preconditions | Click Undo/Redo button |
| Conditions affecting termination outcome |  |

*Table 5 – Use case Change Color description*

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Change Color | | |
| Description | Allows host to change color of the brush | | |
| Actor | Host | | |
| Business event | No. | Agent | System |
|  | 1 | Click Pen button |  |
|  | 2 |  | Open color dialog |
| Preconditions |  | | |
| Conditions affecting termination outcome |  | | |

*Table 6 – Use case Create Room description*

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Create Room | | |
| Description | Create a room for host | | |
| Actor | User | | |
| Business event | No. | Agent | System |
|  | 1 | Click Create button |  |
|  | 2 |  | A connection is established between host and server. |
|  | 3 |  | The server create ID for host and room ID, set host = true and presenter = true. |
|  | 4 |  | The server redirect to the room page |
| Preconditions | Server is running | | |
| Conditions affecting termination outcome | The server is running, the connection is established successfully. The server is terminated, connection failed. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Real-time Chat | | |
| Description | Allow users to communicate with each other | | |
| Actor | Host, viewer | | |
| Business event | No. | Agent | System |
|  | 1 | Enter the message to a text box |  |
|  | 2 | Click Send button |  |
|  | 3 |  | Send the message to the server to broadcast to other people that are  connected to the same artboard |
|  | 4 |  | Display the message to the chatbox |

|  |  |
| --- | --- |
| Preconditions | User is connected to the server |
| Conditions affecting termination outcome |  |

*Table 8 – Use case Join Room description*

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Join Room | | |
| Description | Join a room for viewer | | |
| Actor | viewer | | |
| Business event | No. | Agent | System |
|  | 1 | Click Join button |  |
|  | 2 |  | A connection is established between viewer and server. |
|  | 3 |  | The server create ID for viewer and room ID, set host = false and presenter = false. |
|  | 4 |  | The server redirect to the room page |
| Preconditions | Server is running | | |
| Conditions affecting termination outcome | The server is running, the connection is established successfully. The server is terminated, connection failed. | | |

# Task Assignment

*Table 9 – Work Plan*

|  |  |  |
| --- | --- | --- |
| Student’s name | Evaluate contribution | Task |
| Le Bui Huu Phuc | 100% | Undo mode  Redo mode  Pencil mode  Eraser mode |
| Le Bui Huu Phuc | 100% | Create Room, Join Room Form UI and Controller |
| Le Bui Huu Phuc | 100% | User privilege (host or viewer ) |
| Nguyen Khac Huy | 100% | Change color mode  Shape mode |
| Nguyen Khac Huy | 100% | White Board Form UI and controller |
| Nguyen Khac Huy | 100% | Real-time Chat function |

*Table 10 – Work Assignment*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Building an Remote Draw software using Java | | | | | | | | | | |
| Goal | Schedule | | | | | | | | | |
| Define Requirements | o | o |  |  |  |  |  |  |  |  |
| Set Up Project Structure | o | o |  |  |  |  |  |  |  |  |
| Server-Side Development (Node.js) |  | o | o | o |  |  |  |  |  |  |
| Frontend Development (React.js) |  | o | o | o |  |  |  |  |  |  |
| Implement Collaborative Board Features |  | o | o | o |  |  |  |  |  |  |
| User Privilege |  |  | o | o |  |  |  |  |  |  |
| Testing |  |  | o | o | o |  |  |  |  |  |
| Continuous Improvement |  |  |  |  | o | o | o | o | o |  |
| Testing |  |  |  |  |  |  | o | o | o | o |
| Write report |  |  |  |  |  |  | o | o | o | o |
| Day | 28/09/2023 | 05/10/2023 | 12/10/2023 | 19/10/2023 | 26/10/2023 | 02/11/2023 | 09/11/2023 | 16/11/2023 | 23/11/2023 | 30/11/2023 |
| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Note | * – Begin * – Complete 50% * – Complete 100% | | | | | | | | | |

# Design

## Process description

### Controller protocols

*Table 11 – Controller protocols*

|  |  |  |
| --- | --- | --- |
| **Object** | **Event** | **Explaination** |
| **CreateRoom Form** | handleRoomJoin | This function is responsible for handling the room joining process. When triggered, it sends an event named "userJoined" to the server, containing room data, including name, roomId, userId (randomly generated from the uuid() method declared in App.jsx file), host set to true, and presenter set to true. This event signifies that a user has joined the room. |
| userJoined | Upon receiving the "userJoined" event, the server processes the data and acknowledges the user's presence by sending the event "userIsJoined" back. Additionally, it includes information about the room, such as name, roomId, userId, host status, and presenter status. The user is then navigated to the designated whiteboard room using the navigate(/${roomId}) method. |
| **JoinRoom Form** | handleRoomJoin  userJoined | The JoinRoom Form follows a similar protocol to the CreateRoom Form, with the distinction that the host and presenter attributes are set to false when sending the "userJoined" event to the server. |
| **App Components** | userIsJoined | After the server receives the "userJoined" event from both the CreateRoom Form and JoinRoom Form, it responds by sending the "userIsJoined" event back to the respective clients. This event serves as an announcement, informing the user whether they have successfully joined the room or encountered an error during the process. |
| allUsers | The App component receives the "allUsers" event from the server, containing data about all users present in the room. This information is then combined with the existing "data" state using the useEffect() method. This ensures that the application maintains an updated record of all users within the room. |
| userJoinedMessageBroadcasted & userLeftMessageBroadcasted | These two events are utilized for user announcements. When a user joins or leaves the room, the server broadcasts these events. The App component listens for these events and displays toastify popups, providing real-time notifications of user activities. |
| **Whiteboard Component** | whiteBoardData | The Whiteboard Component sends data to the server with the message "whiteBoardData." The transmitted data includes an image of the elements present on the canvas board. Upon receiving this data, the server broadcasts it to all users (excluding the host). |
| whiteBoarDataResponse | This component listens for the data response of the whiteboard from the server. Upon receiving the response, it updates the data state of the image, ensuring that all users have synchronized whiteboard content. |

### Drawing

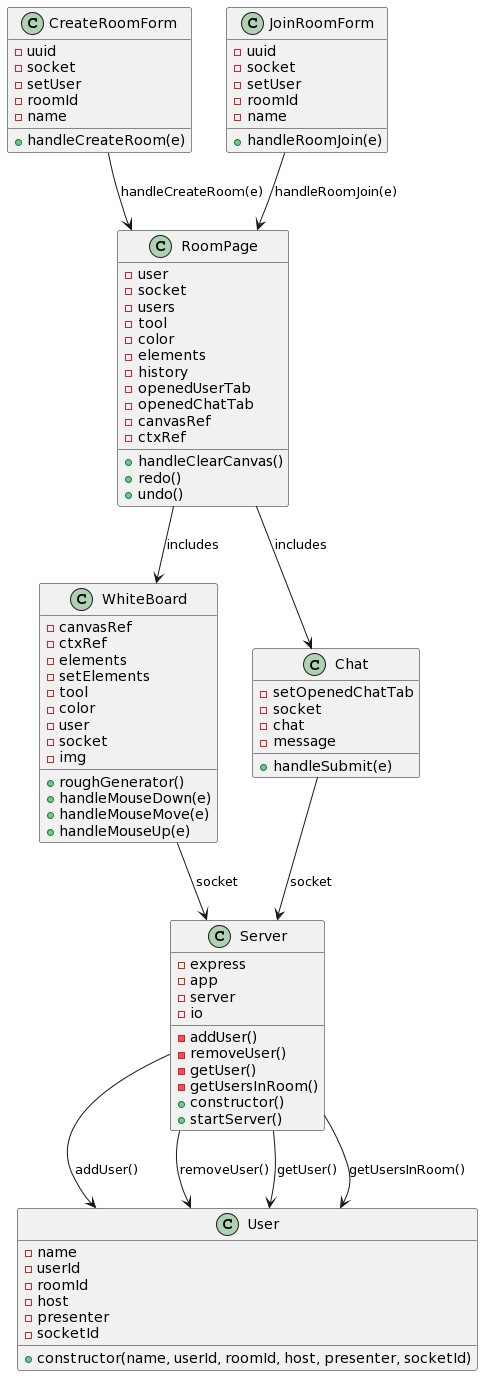
*Table 12 – Drawing method*

|  |  |
| --- | --- |
| Attribute/Function | Explanation |
| ‘const [isDrawing, setIsDrawing] = useState(false);’ | State variable isDrawing is initialized to false, indicating that the user is not currently drawing. |
| 'const handleMouseDown = (e) => {...}' | This function is triggered when the mouse button is pressed. It captures the initial position (offsetX and offsetY) for drawing based on the selected tool (pencil, line, or rectangle). Depending on the tool, a new element is added to the elements state array with relevant properties. |
| ‘const handleMouseMove = (e) => {...}’ | This function is invoked when the mouse is moved. If drawing is in progress (isDrawing is true), it updates the drawing path, width, or height based on the selected tool. For example, when using the pencil tool, it appends the current position to the path; for the line tool, it updates the endpoint coordinates; and for the rectangle tool, it adjusts the width and height. |
| ‘const handleMouseUp = (e) => {...}’ | This function is called when the mouse button is released, signifying the end of the drawing process. It sets isDrawing to false, indicating that the user has finished drawing. |

This drawing logic enables users to interactively create and modify elements on the canvas based on the selected tool, facilitating a dynamic and collaborative drawing experience on the whiteboard.

## Class Design

Below is our overview of class design



*Table 13 – Class Design*

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Properties** | **Methods** | **Relationships** |
| RoomPage | - user: Represents the user associated with the room page.  - socket: Socket connection for real  -time communication.  - users: List of users in the room. - tool: Current drawing tool selected.  - color: Current drawing color selected.  - elements: List of drawing elements on the whiteboard.  - history: History of drawing actions. - openedUserTab: Flag indicating whether the user tab is open.  - openedChatTab: Flag indicating whether the chat tab is open.  - canvasRef: Reference to the whiteboard canvas.  - ctxRef: Reference to the whiteboard canvas context | -handleClearCanvas(): Method to clear the whiteboard. - redo(): Method to redo the last drawing action. - undo(): Method to undo the last drawing action. | Includes WhiteBoard, Chat |
| CreateRoomForm | - uuid: Utility for generating unique IDs.  - socket: Socket connection for real-time communication.  - setUser: Method for setting the user.  - roomId: ID of the room being created.  - name: Name associated with the room. | - handleCreateRoom(e): Method to handle the creation of a room and redirect to the RoomPage. | Redirects to RoomPage |
| JoinRoomForm | - uuid: Utility for generating unique IDs.  - socket: Socket connection for real-time communication.  - setUser: Method for setting the user.  - roomId: ID of the room being joined.  - name: Name associated with the user joining the room. | - handleRoomJoin(e): Method to handle joining a room and redirect to the RoomPage. |  |

## Graphic User Interface

*Table 14 – GUI explanation*

|  |  |  |  |
| --- | --- | --- | --- |
| No. | GUI | Purpose | Brief Explanation |
| 1 | **Room Form**   * *Host :*      * *Viewer :* | The main room window consist of whiteboard and functions | This is the main form of our project which in dividing into two mode ( host and viewer ). The host can completely use the whiteboard while the viewer only can views and chats. |
| 2 | **Chat box** | Communicating | Generally, user can type in the textbox and then press enter to send to others. The message will be displayed as log form. |
| 3 | **Join and Create Room form** | Allow user to create a new room or join an existing room | For creating : user can create a room with a code generated randomly by our format defined function then user will be redirected to main Room with full of control.  For Joining : user after entering the code of existing room, they can join the board room with viewer mode then. |

# Conclusion

## Limitations:

* Minor bugs :
* Refresh the web page can cause lost host control

+ This is because we still haven’t save the session yet -> create a session saver for saving the session of whiteboard that prevent lost host key

* Due to the lack of saving the session, it also lead to error in user online management
* We still not deploy the project on the cloud network.
* The restriction of knowledge about coding also make our project still isn’t optimized ( the some part of code not clean)

## Difficulties

* Learning new technology is a problem for us because it slows down the project progress
* Multi-threading programming is also a problem because we do not have enough knowledge and practice.

## Development ideas

* Increased Line Thickness and Additional Shapes: Enhance the drawing experience by incorporating thicker lines and introducing more shapes to the whiteboard.
* Export Whiteboard Content to PNG: Enable users to export the content on the whiteboard to a PNG format, providing a convenient way to save and share their work.
* Apply CSS for Improved User Interface: Utilize Cascading Style Sheets (CSS) to improve the user interface, applying styles and layouts to enhance the overall visual appeal and user experience.
* Implement User Registration and Login with Database: Establish a database system to facilitate user registration and login functionality. This allows users to create accounts, log in with a username, and securely save their progress.
* Host-Controlled Viewer Permissions on Whiteboard: Introduce a feature where the host has the ability to grant or restrict viewer permissions to draw on the whiteboard. This enhances control and customization over collaborative drawing session.

# References

We developed this project inspired of this guy’s project :

[https: //github.com/RamanSharma100/mern-sockeio-realtime-board-sharing-app](https://github.com/RamanSharma100/mern-sockeio-realtime-board-sharing-app)

Reactjs and Nodejs Documentation :

<https://devdocs.io/react/>

<https://nodejs.org/en/docs>

Socket.io controller Documentaion :

<https://socket.io/docs>

Class Diagram :

<https://www.plantuml.com>