A SYNOPSIS ON

Real-Time Collaborative Whiteboard

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by:

Charanjeet Rawat	**2261152**
Vijay Joshi	**2261601**
Gaurav Joshi	**2261217**
Rajat Bhatt	**2261457**

Under the Guidance of

Mr. Prince Kumar

Assistant Professor

Project Team ID: 49

Department of Computer Science & Engineering

Graphic Era Hill University, Bhimtal, Uttarakhand

March-2025

Graphic Era Hill University
BHIMTAL CAMPUS

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the Synopsis entitled **" Real-Time Collaborative Whiteboard"** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering of the Graphic Era Hill University, Bhimtal campus and shall be carried out by the undersigned under the supervision of **Mr. Prince Kumar, Assistant** , Department of Computer Science & Engineering, Graphic Era Hill University, Bhimtal.

Charanjeet Rawat	2261152	signature
Vijay Joshi	2261601	signature
Gaurav Joshi	2261217	signature
Rajat Bhatt	2261457	signature

The above mentioned students shall be working under the supervision of the undersigned on the **" Real-Time Collaborative Whiteboard"**

Signature

Supervisor

Signature

Head of the Department

Internal Evaluation (By DPRC Committee)

Status of the Synopsis: Accepted / Rejected

Any Comments:

Name of the Committee Members: **Signature with Date**

1.

2.

Table of Contents

Chapter 1

Introduction and Problem Statement

1.1 Introduction

In today's fast-paced digital environment, effective collaboration is essential across various industries, including education, business, software development, and design. Traditional physical whiteboards have been a primary tool for brainstorming, planning, and visualization. However, with the rise of remote work, online learning, and distributed teams, there is an increasing demand for real-time digital whiteboards that allow users to interact seamlessly, regardless of their physical location.

The Real-Time Collaborative Whiteboard project aims to provide an interactive web-based platform that supports multiple users working simultaneously on a shared digital canvas. This whiteboard will include **drawing tools, text input, image uploads, and real-time synchronization** to facilitate efficient collaboration. The project will leverage modern web technologies such as **React, Next.js, Node.js, WebSocket (Socket.IO), MongoDB, JWT, and OAuth** to ensure a secure, scalable, and responsive user experience.

1.1.1 Use Cases

The demand for a real-time digital whiteboard spans various domains:

- **Education**: Teachers can conduct online classes, illustrate concepts, and collaborate with students in real time.
- **Business & Corporate Meetings:** Teams can brainstorm, strategize, and create diagrams without being physically present.
- **Software Development & UI/UX Design:** Developers and designers can collaborate on wireframes, architecture diagrams, and workflows.
- **Creative Work:** Artists, designers, and content creators can use the platform for sketching, ideation, and design discussions.

By offering a feature-rich, real-time collaborative whiteboard, this project aims to enhance remote teamwork, streamline ideation, and improve digital communication.

1.2 Problem Statement

1.2.1 Challenges in Existing Solutions

Despite the availability of some digital collaboration tools, they suffer from various limitations:

1. **Lack of real-time synchronization:** Many existing whiteboards introduce noticeable latency or synchronization delays, making it difficult for multiple users to work simultaneously.

2. **Limited toolset:** Some tools only support basic pen and eraser functions, without additional features like shapes, text formatting, and image manipulation.
3. **High-cost or subscription-based models:** Many effective collaborative tools require expensive subscriptions, making them inaccessible for smaller teams or educational institutions.
4. **Poor integration with existing workflows:** Collaboration tools like Slack, Microsoft Teams, and Zoom are widely used, but most whiteboards do not seamlessly integrate with these platforms.
5. **Security concerns:** Online collaboration platforms often lack proper authentication and role management, leading to unauthorized access and data security risks.
6. **Data persistence issues**: Users often need to save, retrieve, and export their whiteboard sessions for future reference, but many platforms lack efficient storage and retrieval mechanisms.

1.2.2 How This Project Addresses the Problem

The Real-Time Collaborative Whiteboard aims to overcome these challenges by offering a secure, feature-rich, and high-performance solution. The key improvements include:

✓ Real-time, low-latency synchronization powered by WebSocket (Socket.IO) to ensure instant updates.
✓ Comprehensive drawing and text tools, including pen, highlighter, shapes, fonts, and image uploads for enhanced creativity.
✓ Secure user authentication using JWT and OAuth, ensuring that only authorized users can access and modify the whiteboard.
✓ Session persistence with MongoDB, allowing users to save and retrieve whiteboard sessions.
✓ Export options for saving whiteboard content as images (PNG, JPEG) or PDFs for documentation and offline reference.
✓ Integration with third-party tools (e.g., Slack, Microsoft Teams) for seamless collaboration within existing workflows.
✓ Scalable architecture using React and Next.js for a fast, interactive frontend and Node.js with Express for a robust backend**.**

Chapter 2

Background/ Literature Survey

Collaboration tools have evolved significantly in recent years, driven by advancements in real-time web technologies, cloud computing, and digital communication platforms. Traditional whiteboards have been widely used for brainstorming, planning, and idea-sharing, but their physical limitations have led to the development of digital whiteboards that facilitate remote collaboration.

This chapter explores existing solutions, relevant research, and the technological advancements that support the development of a real-time, interactive, and feature-rich digital whiteboard.

2.2 Existing Solutions

Several digital whiteboard applications currently exist, but they have limitations in terms of real-time synchronization, toolset diversity, security, and accessibility. Some of the most popular existing platforms include:

2.2.1 Microsoft Whiteboard

Strengths:

- Seamless integration with Microsoft Teams.
- Supports inking, text, and image uploads.
- Allows real-time collaboration with cloud storage.

Limitations:

- Requires Microsoft 365 subscription for full functionality.
- Limited third-party integrations (does not support Slack).
- Lag issues with multiple users working simultaneously.

2.2.2 Google Jamboard

Strengths:

- Integrated with Google Workspace (Docs, Drive, Meet).
- Supports basic drawing, text input, and sticky notes.
- Cloud-based with multi-user collaboration.

Limitations:

- Limited drawing tools (no highlighter, limited shape tools).
- No session persistence (does not save full history of interactions).
- Requires a Google account, making external collaborations difficult.

2.2.3 Miro

Strengths:

- Supports real-time collaboration, diagramming, and sticky notes.
- Integration with third-party tools like Slack, Trello, and Jira.
- Allows team-based workspaces for better organization.

Limitations:

- Expensive premium plans for extended features.
- Performance issues with a large number of users.
- Limited customization compared to open-source solutions.

2.2.4 Open-Source Whiteboards (Excalidraw, Draw.io, tldraw)

Strengths:

- Free and open-source, allowing customization.
- Lightweight and easy to integrate into web applications.

Limitations:

- Lack of real-time collaboration features (most are single-user focused).
- No built-in authentication or session storage**.**
- Limited cloud-based integration options.

2.3 Research Trends

- **WebSocket-Based Synchronization:** Reduces latency in real-time applications**.**
- **Conflict Resolution Techniques:** Uses Operational Transformation (OT) and Conflict-Free Replicated Data Types (CRDTs) to handle concurrent edits**.**
- **Security Enhancements**: JWT authentication and OAuth integration ensure secure access.

2.4 Gap Analysis

Current whiteboards lack real-time efficiency, security, and integration capabilities. This project fills these gaps by implementing WebSocket-based updates, comprehensive toolsets, and enhanced authentication mechanisms**.**

Chapter 3

Objectives

The **Real-Time Collaborative Whiteboard** project aims to provide an **efficient, secure, and feature-rich** platform for real-time collaboration. The following objectives outline the key goals of the project:

1.Develop a Real-Time, Low-Latency Whiteboard

- Implement **WebSocket (Socket.IO)** to enable **instant synchronization** between multiple users.
- Ensure **smooth drawing, text input, and image manipulation** without noticeable delay.

2.**Integrate a Comprehensive Set of Drawing and Editing Tools**

- Provide **pen, highlighter, shape tools, text boxes, and image uploads** for enhanced collaboration.
- Allow users to **resize, move, and modify elements** on the whiteboard.

3. **Implement Secure User Authentication and Role Management**

- Use **JWT (JSON Web Tokens) and OAuth** for **user authentication**.
- Implement **role-based access control (RBAC)** to manage **admins, editors, and viewers**.

4.**Enable Session Management and Whiteboard Persistence**

- Allow users to **save, load, and resume whiteboard sessions**.
- Provide options to **export whiteboards as images (PNG, JPEG) or PDFs** for offline use.

5.**Ensure Scalability and Integration with Other Collaboration Tools**

- Optimize the system to support **multiple users without performance degradation**.
- Integrate with platforms like **Slack, Microsoft Teams, and Google Drive** for seamless workflow adoption.

By achieving these objectives, the project will provide an **interactive, secure, and scalable digital whiteboard** that enhances **remote teamwork, education, and brainstorming sessions**.

Chapter 4

Hardware and Software Requirements

4.1 Hardware Requirements

Sl. No	Name of the Hardware	Specification
1	Server	Cloud-based
2	Client Devices	PC,Tablet,Moblies

4.2 Software Requirements

Sl. No	Name of the Software	Specification
1	Operating System	Windows 10/11, macOS, or Linux
2	Frontend Framework	React, Next.js
3	Backend Framework	Node.js with WebSocket (Socket.IO), Express
4	Database	MongoDB
5	Authentication	JWT, OAuth

Chapter 5

Possible Approach/ Algorithms

Developing a real-time collaborative whiteboard requires a combination of synchronization algorithms, networking protocols, and efficient data handling mechanisms to ensure seamless multi-user collaboration. Below, we explore various approaches and algorithms used in the project.

5.1 Real-Time Data Synchronization

A real-time whiteboard must synchronize multiple users' actions (drawing, erasing, adding text, etc.) across different devices with minimal latency. The following approaches are commonly used:

5.1.1 WebSockets for Bidirectional Communication

- **Protocol Used:** *WebSocket (via Socket.IO)*
- **Why WebSockets?**
    - Unlike HTTP (which is request-response based), WebSockets enable a persistent connection between the server and clients, allowing real-time updates without polling.
    - This ensures low-latency collaboration by broadcasting changes instantly.
- **Implementation:**
    - A Socket.IO server maintains connections and listens for user events (drawing, erasing, text input, etc.).
    - Each update is broadcast to all connected clients using WebSocket events.

5.2 Concurrency Control and Conflict Resolution

Since multiple users can modify the whiteboard simultaneously, conflict resolution strategies are required:

5.2.1 Operational Transformation (OT) Algorithm

- **Concept:**
    - Used in Google Docs, OT ensures consistent state across all users when concurrent edits occur.
- **How It Works:**
    - Each operation (draw, erase, text) is timestamped and transformed if conflicts arise.

- **The transformation ensures that all clients apply changes in the correct order.**

5.2.2 Conflict-Free Replicated Data Types (CRDTs)

- **Concept:**
  - Unlike OT, CRDTs do not require a central server to resolve conflicts.
  - It ensures eventual consistency by merging updates locally without overwriting data.
- **Use Case:**
  - Useful for peer-to-peer or decentralized whiteboard applications.

5.3 Stroke Simplification and Compression

Since real-time drawing involves a large volume of data (continuous mouse movements), optimizing stroke data is crucial.

5.3.1 Douglas-Peucker Algorithm *(Polyline Simplification)*

- **Purpose:**
  - Reduces the number of points in a stroke while maintaining shape accuracy.
- **How It Works:**
  - Identifies critical points in a stroke and removes redundant intermediate points.
- **Advantage:**
  - Reduces network bandwidth usage without compromising drawing quality.

5.3.2 Run-Length Encoding (RLE) for Compression

- **Concept:**
  - Consecutive similar stroke points (same color, thickness, opacity) are encoded as a single value instead of storing each separately.
- **Outcome:**
  - Lowers memory usage and improves performance.

5.4 Undo/Redo Mechanism

5.4.1 Stack-Based Undo/Redo

- **Approach:**
  - **Uses two stacks:**
    - **Undo Stack:** Stores previous states of the whiteboard.

- **Redo Stack:** Stores undone actions for re-execution.

- **Operations:**
  - ***Undo*:** Pops the last state from the undo stack and pushes it to the redo stack.
  - ***Redo*:** Pops from redo stack and applies it back.

5.4.2 Command Pattern Approach

- **Concept:**
  - Every action (draw, erase, text) is treated as a command object.
- **Advantage:**
  - Ensures better modularity and extensibility.

5.5 Data Storage and Persistence

5.5.1 JSON-Based Data Structure

- **Reason:**
  - The whiteboard elements (shapes, strokes, text) are stored as JSON objects, making it easy to serialize/deserialize data.
- **Example:**
- {
- "type": "pen",
- "points": [[12, 15], [14, 16], [18, 19]],
- "color": "#ff0000",
- "thickness": 2
- }

5.5.2 MongoDB for Storing Whiteboard Sessions

- **Why MongoDB?**
  - NoSQL format allows efficient storage of dynamic whiteboard data.
- **Implementation:**
  - Whiteboard session data is saved periodically to allow users to resume work later.

5.6 Exporting Whiteboards as Images/PDFs

5.6.1 HTML5 Canvas API for Image Export

- **Approach:**
    - The Canvas API captures the entire whiteboard and converts it into an image format (*PNG, JPEG*).
- **Implementation:**

```
const canvas = document.getElementById('whiteboard');
const image = canvas.toDataURL('image/png');
```

5.6.2 jsPDF Library for PDF Export

- **Approach:**
    - The jsPDF library captures the whiteboard content and saves it as a PDF file.
- **Implementation:**

```
const pdf = new jsPDF();
pdf.addImage(image, 'PNG', 10, 10);
pdf.save("whiteboard.pdf");
```

5.7 Authentication and User Roles

5.7.1 JWT (JSON Web Token) for Secure Authentication

- **Why JWT?**
    - Stateless authentication method that encodes user sessions securely.
- **Implementation:**
    - Upon login, the server issues a JWT token, which is stored in the client's browser for subsequent API calls.

5.7.2 OAuth 2.0 for Third-Party Authentication

- **Why OAuth?**
    - Allows users to sign in using Google, GitHub, or Microsoft, reducing login friction**.**
- **Implementation:**
    - The backend verifies OAuth tokens before granting access.

5.8 Integration with Other Collaboration Tools

To make the whiteboard more useful for teams, integration with external tools like Slack or Microsoft Teams is needed.

5.8.1 Webhooks for Real-Time Notifications

- **Concept:**
    - When a whiteboard session is updated, a Webhook notification is sent to Slack/Microsoft Teams**.**

5.8.2 API-Based Integration

- **Approach:**
    - The backend exposes RESTful APIs that external tools can fetch updates from.

5.9 Voice-to-Text Input

To enhance accessibility and user convenience, voice input can be integrated into the whiteboard, allowing users to dictate text instead of manually typing.

5.9.1 Web Speech API for Voice Recognition

- **Why Web Speech API?**
    - Built-in support in modern browsers (Chrome, Edge).
    - No need for external dependencies.
- **Implementation:**

```
const recognition = new webkitSpeechRecognition() || new SpeechRecognition();
recognition.continuous = true;
recognition.interimResults = true;
recognition.onresult = (event) => {
   const transcript = event.results[event.results.length - 1][0].transcript;
   document.getElementById("textbox").value = transcript;
};
recognition.start();
```

- **How It Works?**
    - The API listens to user speech and converts it into text, which is then inserted into a text box on the whiteboard.

References

Ian Pointer, *Programming WebRTC: Build Real-Time Streaming Applications for the Web*, O'Reilly, 2021.

MDN Web Docs, "Using WebSockets" – https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

Fraser, C., & De Moura, L. (2020). *Conflict-Free Replicated Data Types (CRDTs)* – https://crdt.tech/

Sun, C., & Ellis, C. (1998). "Operational transformation in real-time group editors." ACM Transactions on Computer-Human Interaction (TOCHI).

Douglas, D. H., & Peucker, T. K. (1973). "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature." The Canadian Cartographer.

Ramer, U. (1972). "An iterative procedure for the polygonal approximation of plane curves." Computer Graphics and Image Processing.

J. Bradley & N. Sakimura, *OAuth 2.0: Getting Started in Web Security*, 2019.

Official JWT Documentation – https://jwt.io/introduction/

Rabiner, L. (1989). "A tutorial on hidden Markov models and selected applications in speech recognition." Proceedings of the IEEE.

React & Next.js Documentation – https://react.dev/ | https://nextjs.org/

MongoDB Documentation – https://www.mongodb.com/docs/manual/