**INTERACTIVE COLLABORATIVE WHITEBOARD**



**Submitted By**

Sheheryar Salman

22k-4248

Ahmed Raza

22k-4422

# Department of Computer Science

**National University of Computer & Emerging Sciences**

## 1. Motivation

In today’s fast-paced digital environment, real-time collaboration has become essential for productivity and innovation. This project was chosen to tackle the increasing demand for seamless and secure multi-user online collaboration tools, focusing on optimizing performance over varying network conditions and ensuring a smooth, fault-tolerant user experience.

## 2. Overview

### 2.1 Significance of the Project

This project holds high importance due to the growing trend of remote work and online education. Traditional whiteboards lack the collaborative features needed in distributed environments. Our solution is significant for educational, corporate, and creative fields. It’s a technically challenging project requiring real-time data exchange, synchronization, and performance optimization, making it a valuable academic undertaking.

### 2.2 Description of the Project

The Interactive Collaborative Whiteboard is a browser-based application that enables multiple users to draw and annotate simultaneously on a shared canvas in real-time. It utilizes Socket.IO for low-latency data transfer and Protocol Buffers for optimized data transmission. The backend supports user reconnection with session persistence, and the server is designed to handle concurrent users efficiently, making the platform robust and scalable.

### 2.3 Background of the Project

This project builds upon concepts from computer networking, real-time systems, and collaborative software design. References include:

* <https://socket.io/docs/v4/>
* <https://fabricjs.com/docs/>
* <https://supabase.com/docs>
* <https://mui.com/material-ui/getting-started/>

### 2.4 Project Category

Product-Based

## 3. Features / Scope / Modules

1. **Real-Time Drawing Synchronization**
   * Users can draw on a shared canvas simultaneously with near-instant updates via Socket.IO.
2. **Optimized Data Transmission**
   * Use of Protocol Buffers to encode drawing data in a lightweight, binary format, reducing bandwidth.
3. **Session Persistence & Recovery**
   * Users can reconnect to the same session with their data intact using state management on the server.
4. **Concurrency management**
   * Server-side logic includes mechanisms to manage and handle concurrent users efficiently.
5. **Collaborative User Interface**
   * Clean, user-friendly interface designed using Fabric.js to enable intuitive interactions.

## 4. Project Planning

|  |  |  |
| --- | --- | --- |
| Week | Tasks (Ahmed Raza) | Tasks (Sheheryar Salman) |
| 1 | UI Setup & Socket.IO | WebSocket Server Setup & API Design |
| 2 | Sync Optimization | Implement REST routes |
| 3 | UI State Handling | Concurrency management |
| 4 | Reconnection Logic | Session Management Integration |
| 5 | Testing & Bug Fixes | Testing & Bug Fixes |

## 5. Project Feasibility

* **Technical Feasibility**

All technologies used (Node.js, Socket.IO, Protocol Buffers) are open-source and well-documented. Potential technical challenges include reconnection handling and performance tuning, which will be addressed via testing and modular design.

* **Economic Feasibility**

The project requires minimal financial investment as it is developed using open-source tools. Hosting may require a basic cloud service such as Hostinger or Versal.

* **Schedule Feasibility**

The project has been planned across 5 weeks with clear responsibilities assigned to each member, ensuring timely delivery.

## 6. Hardware and Software Requirements

* **Hardware**
  + Personal Computers with Internet Access
  + Server Hosting (e.g., VPS, Heroku for deployment)
* **Software**
  + Node.js, Express.js
  + Socket.IO
  + Protocol Buffers
  + Fabric.js
  + Git for version control
  + Web browser (for testing)

## 7. Diagrammatic Representation of the Overall System

User A’s Browser

User B’s Browser

Socket.IO Server

Session Management

Concurrency Control and Server Logic

## 8. References

[1] Socket.IO Documentation: <https://socket.io/docs/>  
[2] Protocol Buffers Documentation: <https://protobuf.dev/>  
[3] Fabric.js: <http://fabricjs.com/>  
[4] Supabase Documentation: <https://supabase.com/docs>  
[5] Material UI Documentation: <https://mui.com/>  
[6] Stack Overflow – Developer Discussions and Solutions: <https://stackoverflow.com/>  
[7] React Documentation: <https://react.dev/>  
[8] Node.js Documentation: [https://nodejs.org/en/docs](https://nodejs.org/en/docs%20)