A

Project Report

On

Real-Time Collaborative Whiteboard

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

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

Under the Guidance of

Mr. Prince Kumar

ASSISTANT / ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI,DISTRICT- NAINITAL-263132

2024-2025

STUDENT'S DECLARATION

We, **Charanjeet Rawat**, **Vijay Joshi, Gaurav Joshi**, **Rajat Bhatt** hereby declare the work, which is being presented in the project, entitled ' **Real-Time Collaborative Whiteboard**' in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (B.Tech.)** in the session **2024-2025**, is an authentic record of my work carried out under the supervision of **Mr. Prince Kumar, Assistant Professor**.

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date:

Charanjeet Rawat

Vijay Joshi

Gaurav Joshi

Rajat Bhatt

CERTIFICATE

The project report entitled " Real-Time Collaborative Whiteboard " being submitted by Charanjeet Rawat S/o Mr. Shivendra Singh (2261152) , Vijay Joshi s/o Mr. Girish Chandra Joshi (2261601) , Gaurav Joshi s/o Mr. Lalit Chandra Joshi (2261217) , Rajat Bhatt s/o Mr. Bhatt Ramesh Bishen Dutt (2261457) of B.Tech.(CSE) to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.

Mr. Prince Kumar
(Project Guide)

Dr. Ankur Singh Bisht
(Head, CSE)

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director **'Prof. (Col.) Anil Nair (Retd.)',** GEHU Bhimtal Campus to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that we need. We again want to extend thanks to our president **'Prof. (Dr.) Kamal Ghanshala'** for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to **'Dr. Ankur Singh Bisht'** (Head, Department of Computer Science and Engineering, GEHU Bhimtal Campus), our project guide **'Mr. Prince Kumar'** (Assistant Professor, Department of Computer Science and Engineering, GEHU Bhimtal Campus) and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this report.

Finally, yet importantly, We would like to express my heartiest thanks to our beloved parents, for their moral support, affection, and blessings. We would also like to pay our sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this project.

Charanjeet Rawat , 2261152

Vijay Joshi , 2261601

Gaurav Joshi , 2261217

Rajat Bhatt , 2261457

Abstract

In the era of remote work, digital education, and globally distributed teams, real-time collaboration has become a cornerstone of modern productivity. Traditional whiteboards, while effective in physical spaces, fall short in addressing the needs of remote or hybrid environments.

This project, titled "Real-Time Collaborative Whiteboard," aims to bridge that gap by developing an interactive, web-based whiteboard platform that enables seamless, synchronous collaboration among multiple users across different locations.

The platform supports real-time drawing, text input, shape creation, and image uploading, all rendered on a shared canvas using modern frontend frameworks like React.js and Next.js.

For real-time communication, the system utilizes WebSocket (Socket.IO), ensuring low-latency synchronization between connected clients. To manage user authentication and session security, the project implements JWT (JSON Web Tokens) and OAuth 2.0, allowing both email-based and third-party logins through providers like Google or GitHub.

A Node.js backend, coupled with Express.js, handles server-side logic and API endpoints, while MongoDB is used as the primary data store to persist whiteboard sessions, enabling users to save, load, and export their collaborative work.

Optimizations such as the Douglas-Peucker algorithm and Run-Length Encoding (RLE) are incorporated to reduce stroke data size and enhance performance. The whiteboard also features an intuitive undo/redo mechanism and supports exporting sessions as PNG, JPEG, or PDF using the HTML5 Canvas API and jsPDF library.

Furthermore, the application supports voice-to-text input using the Web Speech API, improving accessibility for users with disabilities or those who prefer dictation. Integration with collaboration tools like Slack and Microsoft Teams is achieved through webhooks and REST APIs, making it a versatile solution for education, business meetings, design sessions, and creative brainstorming.

Overall, this project delivers a robust, scalable, and secure collaborative tool tailored for today's dynamic digital ecosystem, addressing the limitations of existing whiteboard solutions through innovative design and modern web technologies.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Prologue

In the modern era of digital transformation, seamless communication and collaboration have become fundamental pillars for productivity and innovation. As teams increasingly operate across different geographic locations, traditional collaboration methods—such as in-person meetings and physical whiteboards—are no longer sufficient. The need for robust digital alternatives that support real-time interaction has become more apparent, especially in the wake of widespread adoption of remote work, online education, and hybrid workplace environments.

This project, titled **"Real-Time Collaborative Whiteboard,"** addresses this necessity by offering a feature-rich, interactive web-based platform that enables multiple users to collaborate simultaneously on a shared digital canvas. Whether for brainstorming sessions, diagramming, educational illustrations, or team planning, this whiteboard empowers users to communicate ideas visually and effectively from anywhere in the world.

1.2 Background and Motivations

The motivation for developing this project stems from the increasing reliance on remote tools for communication, education, and teamwork. Existing platforms like Microsoft Whiteboard, Google Jamboard, and Miro provide partial solutions but often fall short in key areas such as customization, real-time performance, affordability, and accessibility.

Our team observed that many educational institutions, startups, and freelance groups require reliable digital collaboration tools without the burden of costly subscriptions. Moreover, the lack of open-source platforms with built-in authentication, user roles, and session management highlighted a clear opportunity for innovation.

We were inspired to create a platform that:

- Supports instantaneous multi-user drawing and editing
- Provides a rich set of drawing and interaction tools
- Ensures data security and role-based access
- Integrates smoothly into existing digital workflows

Our whiteboard is not only a tool but a solution to bridge the gap between physical brainstorming and digital collaboration, enhancing how teams think, plan, and create together.

1.3 **Problem Statement**

Despite the availability of several digital whiteboarding applications, current offerings are plagued by notable limitations:

1. High latency and poor synchronization: Users often experience delays in updates, which disrupts the collaborative process.
2. Minimal toolsets: Many platforms only offer basic drawing tools with no support for shapes, image uploads, text formatting, or erasers.
3. Expensive pricing models: Leading whiteboard tools require premium subscriptions, which can be restrictive for students, educators, and small teams.
4. Security and access issues: Lack of robust authentication can lead to unauthorized access or misuse of whiteboards.
5. No session persistence: Most platforms fail to allow users to save and resume sessions or export whiteboards for offline use.
6. Poor integration with third-party tools: Applications like Slack, Microsoft Teams, or Google Drive are often unsupported or difficult to connect.

This project aims to overcome these challenges by developing a real-time collaborative whiteboard that is secure, scalable, affordable, and technically sound.

1.4 **Objectives and Research Methodology**

Project Objectives:

- To create a web-based platform that allows multiple users to interact on the same whiteboard in real time.
- To support drawing tools such as pen, eraser, shapes, highlighters, text, and image uploads.
- To ensure real-time low-latency communication using WebSocket (Socket.IO).
- To implement secure login and user management using JWT and OAuth.
- To enable saving, resuming, and exporting whiteboard sessions in image (PNG/JPEG) and PDF formats.
- To provide voice-to-text functionality for accessibility.
- To allow integration with third-party tools like Slack or Microsoft Teams using webhooks and REST APIs.

Research Methodology:

To achieve the above objectives, we followed a structured research and development process:

1. Literature Review: We analyzed various existing digital whiteboard platforms and identified key gaps.
2. Requirement Gathering: Based on real-world use cases and user expectations, we defined the system specifications.
3. Technology Evaluation: We selected modern and open-source tools (React, Node.js, MongoDB, WebSocket) for the tech stack.
4. Design & Prototyping: We designed UI wireframes and backend architecture.
5. Implementation: We developed the frontend and backend components in iterations.
6. Testing & Optimization: We performed unit testing, integration testing, and user testing to ensure stability, performance, and usability.
7. Documentation & Deployment: We documented the system and hosted it on a cloud server for final review.

1.5 **Project Organization**

The report is structured into several chapters to cover all phases of the project development life cycle:

- Chapter 1: Introduction – Describes the motivation, background, and objectives of the project.
- Chapter 2: Hardware and Software Requirements – Lists the technical specifications needed for both development and deployment.
- Chapter 3: Coding of Functions – Explains key modules, implementation logic, and core functionalities of the system.
- Chapter 4: Snapshots – Displays screenshots of the application's user interface, features, and sample sessions.
- Chapter 5: Limitations – Discusses the current challenges or boundaries faced by the system.
- Chapter 6: Enhancements – Lists the future improvements or new features planned for the system.
- Chapter 7: Conclusion – Provides a closing summary and the overall impact of the project.
- References : Official Documentation , Technical Articles and Papers , Websites.

CHAPTER 2 : PHASES OF SOFTWARE DEVELOPMENT CYCLE

HARDWARE AND SOFTWARE REQUIREMENTS

2.1Hardware Requirement

A. Server-Side Requirements:

Component	Specification
Processor	Multi-core CPU (e.g., Intel Xeon or AMD Ryzen)
RAM	Minimum 8 GB (Recommended: 16 GB)
Storage	Minimum 100 GB SSD
Internet Connection	Stable high-speed broadband (min 20 Mbps)
Hosting Platform	Cloud hosting (e.g., AWS, Heroku, Vercel)

B. Client-Side Requirements:

Device Type	Minimum Specification
PC/Desktop	Modern browser (Chrome, Edge, Firefox), 4 GB RAM
Laptop	Same as above
Tablet	Chrome/Edge with touch support, 3 GB RAM
Mobile Phone	Modern browser, at least 2 GB RAM

2.2 Software Requirement:

The software stack used in this project includes a full JavaScript-based ecosystem with powerful open-source technologies for both frontend and backend development.

A. Operating Systems Supported

- Windows 10 or later
- macOS 11 or later
- Ubuntu 20.04+ or equivalent Linux distributions

B. Frontend Development:

Component	Technology/Tool
UI Framework	React.js
Routing & SSR	Next.js
Styling	Tailwind CSS (or CSS3)
Canvas Drawing	HTML5 Canvas API
Voice Input	Web Speech API

C. Backend Development:

Component	Technology/Tool
Server Runtime	Node.js
Framework	Express.js
Real-Time Sync	WebSocket (Socket.IO)
Authentication	JSON Web Token (JWT)
OAuth Login	Google/GitHub OAuth 2.0

D. Database:

Type	Technology
NoSQL	MongoDB

E. Additional Libraries and APIs:

Purpose	Tool/Library
PDF Export	jsPDF
Image Export	HTML5 Canvas API
Stroke Simplification	Douglas-Peucker Algorithm (Custom JS)
Data Compression	Run-Length Encoding (RLE)
Undo/Redo Implementation	Custom Stack Logic
API Integration	REST API, Webhooks

2.3 Development Tools:

Tool	Purpose
Visual Studio Code	Code editor
Postman	API testing
Git & GitHub	Version control and collaboration
Vercel/Heroku	Hosting frontend/backend
MongoDB Atlas	Cloud-hosted database
Figma	UI/UX design and prototyping

Chapter 3: Coding of Functions

The implementation of the Real-Time Collaborative Whiteboard relies on a modular architecture using modern web technologies including React.js, Node.js, Express.js, MongoDB, and Socket.IO. This chapter explains the core functions and their significance within the system, emphasizing how each module contributes to real-time interaction, secure collaboration, and persistent session management. The system has been divided into front-end and back-end modules for scalability and ease of maintenance.

3.1 Server Initialization and Real-Time Communication

The server is built using Node.js and utilizes Socket.IO to enable real-time, bidirectional communication between connected users. The server listens for incoming WebSocket connections on a specified port. When a user connects, a unique socket ID is assigned, and the connection is managed independently, enabling simultaneous collaboration by multiple users.

Each action performed by a user (drawing, erasing, adding text or images) is captured as an event and broadcasted to all other connected clients in real-time using the WebSocket protocol. This low-latency architecture ensures that all collaborators see live updates instantly, providing a smooth and synchronized whiteboarding experience.

3.2 Request Routing and Session Management

The backend uses Express.js to define RESTful routes for session management, authentication, and database interactions. Static frontend files (built from React) are served from a public/ directory, while dynamic API routes handle user authentication, saving/loading whiteboard sessions, and role-based access control.

For example:

/api/login handles user login requests.

/api/save-session stores whiteboard data to MongoDB.

/api/load-session/:id retrieves saved session data.

Session data, including user ID, role, and token, is verified using JWT and stored temporarily in memory or local storage on the client side. This session data is used to control access and permissions within collaborative sessions.

3.3 Real-Time Whiteboard Interaction (Canvas and WebSocket Integration)

The whiteboard canvas is built using the HTML5 Canvas API, integrated into the React front-end. All mouse and touch inputs are captured and converted into JSON-based stroke data, which is emitted via WebSocket to the server and rebroadcasted to other clients.

Each stroke includes:

Coordinates (x, y)

Color

Tool (pen, highlighter, eraser)

Stroke width

To optimize bandwidth, data is compressed using Run-Length Encoding (RLE), and strokes are simplified using the Douglas-Peucker algorithm before transmission.

3.4 User Authentication and Role-Based Access Control

Secure authentication is implemented using JWT (JSON Web Tokens), while third-party sign-ins use OAuth 2.0 (Google and GitHub). When a user logs in:

A token is generated by the backend and sent to the frontend.

This token is stored in the browser and used for all protected API requests.

Tokens are verified using middleware to ensure authenticity.

Users are assigned roles such as "Admin", "Editor", or "Viewer", which control their access level to certain whiteboard features (e.g., only editors can draw; viewers are read-only).

3.5 Whiteboard Persistence and Export Features

To ensure that users can save their work and continue later, the whiteboard session is periodically saved to MongoDB. Each session stores:

User ID

Canvas data (strokes, shapes, images)

Timestamp

Session title or metadata

Users can export their whiteboard in various formats:

PNG/JPEG using canvas.toDataURL()

PDF using jsPDF, which embeds the canvas image into a PDF document and prompts download

These exports allow easy sharing, documentation, and offline reference of collaborative work.

3.6 Voice Input using Web Speech API

To improve accessibility, the system integrates the Web Speech API. Users can dictate text using voice, which is transcribed and automatically added to the whiteboard as text annotations. This is particularly useful for visually impaired users or those who prefer speech input.

3.7 Admin Dashboard and User Management

Administrators have access to a separate dashboard where they can:

View all registered users.

Inspect and delete sessions.

Assign or change roles.

Monitor collaborative activities.

All actions are protected using JWT verification and role-based middleware. The admin module is key to maintaining user control and collaborative integrity, especially in shared or public environments.

3.8 Error Handling and Logging

The application implements structured error handling on both client and server sides. For example:

Invalid login returns HTTP 401 Unauthorized.

Invalid routes return HTTP 404 Not Found.

Server errors return HTTP 500 Internal Server Error.

Additionally, server-side logs record all critical events including:

Connection/disconnection

API usage

Session save/load status

Authentication failures

Logs are stored with timestamps and IPs to support monitoring and debugging.

3.9 Concurrency and Scalability

Thanks to WebSocket and non-blocking I/O in Node.js, the application handles multiple users concurrently with minimal performance impact. Socket.IO ensures that even with 10+ users in the same session, all interactions are synchronized in real time.

Scalability is further supported by:

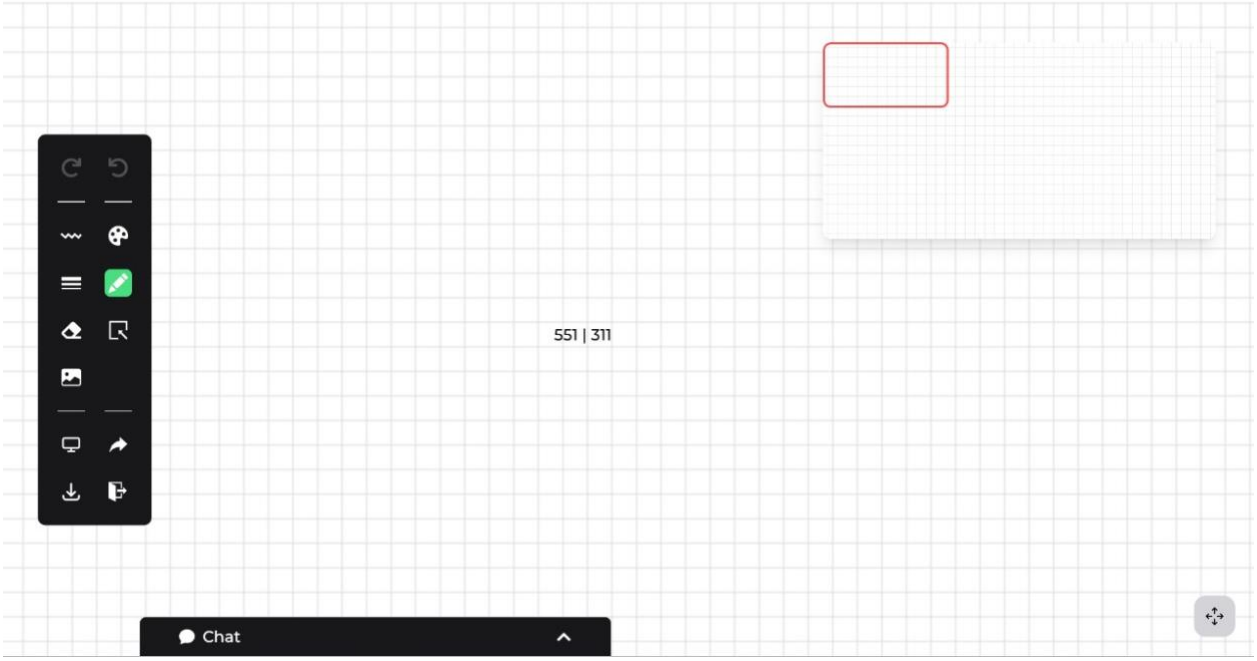
Cloud deployment (e.g., Vercel or Heroku)

MongoDB Atlas for distributed storage

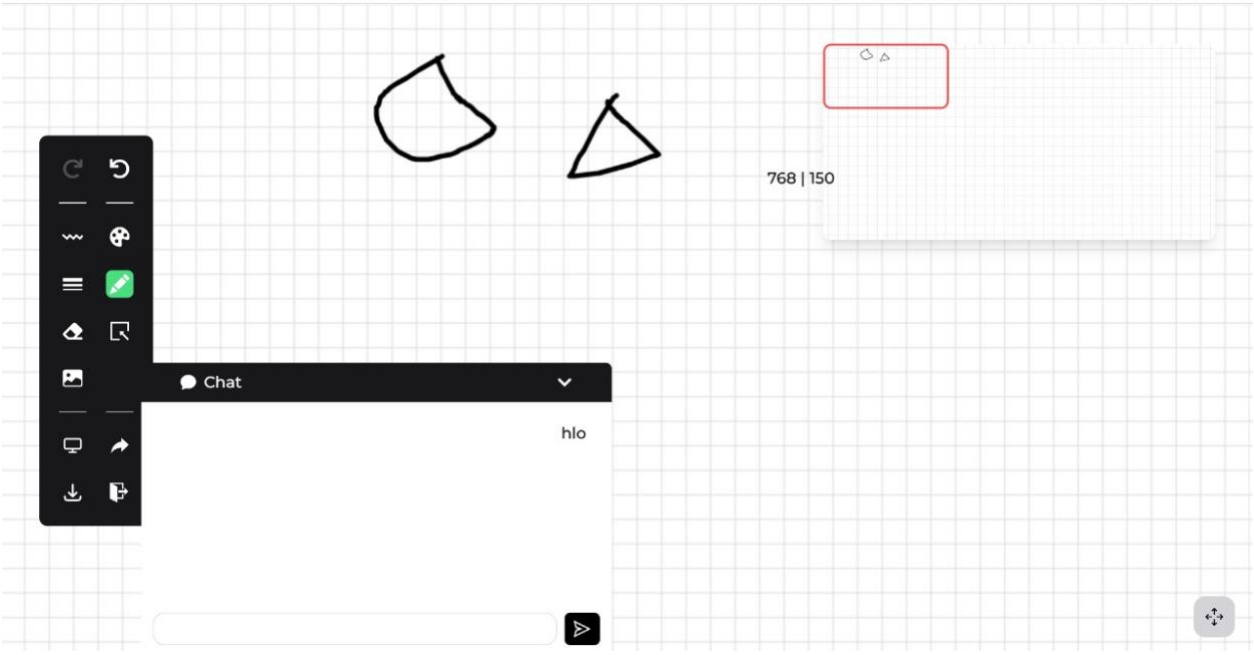
Stateless authentication using JWT

SNAPSHOTS

3.1



3.2



LIMITATIONS :

Despite implementing a wide range of features, the current version of the Real-Time Collaborative Whiteboard has some limitations, as outlined below:

- Limited Offline Support: The application requires a stable internet connection for all collaborative features. Offline usage is currently not supported.

- Mobile Responsiveness: Although the application works on tablets and mobile phones, the user interface is best optimized for desktops and may need additional tweaks for small screens.

- Limited Storage Duration: Saved sessions are stored in MongoDB but may be purged after a certain time due to storage limitations in free-tier cloud hosting.

- Basic Access Control: Role-based access is implemented, but fine-grained control (e.g., per-object or per-layer permissions) is not yet available.

- No Version History: Users can save and export sessions, but cannot view or restore earlier versions from a timeline/history feature.

ENHANCEMENTS:

To further improve the application and meet broader user needs, the following

enhancements are planned for future development:

1. **Mobile App Version:** Developing native Android and iOS versions for better mobile experience and offline editing.

2. **Mobile App Version:** Developing native Android and iOS versions for better mobile experience and offline editing

3. **Audio/Video Integration:** Integrate WebRTC-based video/audio conferencing for complete remote teamwork support.

4. **Version Control:** Implement whiteboard history and version rollback for better tracking of changes over time.

5. **Layered Canvas:** Introduce layers for better object organization and design workflows.

6. **Export to SVG:** Add support for exporting whiteboard content in vector (SVG) format for designers and high-quality printing.

7. **AI Features:** Add handwriting-to-text and shape correction using basic machine learning models or browser-based AI.

8. **Multi-language Voice Input:** Extend Web Speech API support to multiple languages and dialects for global users.

CONCLUSION:

The Real-Time Collaborative Whiteboard project successfully addresses the growing need for an interactive, accessible, and scalable collaboration platform in modern digital environments. Through real-time synchronization, secure authentication, and a feature-rich drawing interface, this application empowers users to share ideas visually from anywhere in the world.

By leveraging open-source technologies like React, Node.js, Socket.IO, and MongoDB, we have created a solution that is technically robust and financially viable for teams, educators, and creatives. Though limitations still exist, the proposed enhancements provide a clear path for evolving the system into a fully-fledged digital workspace. Overall, the project demonstrates the power of modern web technologies in transforming traditional collaboration tools into intelligent, connected platforms.

REFERENCES:

1. Pointer, I. (2021). Programming WebRTC: Build Real-Time Streaming Applications for the Web. O'Reilly Media.
2. Sun, C., & Ellis, C. (1998). Operational Transformation in Real-Time Group Editors. ACM Transactions on Computer-Human Interaction (TOCHI), 5(1), 63–108.
3. Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line. The Canadian Cartographer, 10(2), 112–122.
4. Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77(2), 257–286.
5. Mozilla Developer Network (MDN). (n.d.). Using WebSockets. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
6. CRDT.tech. (n.d.). Conflict-Free Replicated Data Types (CRDTs). Retrieved from https://crdt.tech/
7. JSON Web Tokens (JWT). (n.d.). Introduction to JSON Web Tokens. Retrieved from https://jwt.io/introduction/
8. React.js. (n.d.). React Documentation. Meta. Retrieved from https://react.dev/learn
9. Next.js. (n.d.). Next.js Documentation. Vercel. Retrieved from https://nextjs.org/docs
10. Node.js. (n.d.). Node.js Official Documentation. OpenJS Foundation. Retrieved from https://nodejs.org/en/docs
11. Socket.IO. (n.d.). Socket.IO Documentation. Retrieved from https://socket.io/docs/v4
12. MongoDB. (n.d.). MongoDB Manual. MongoDB Inc. Retrieved from https://www.mongodb.com/docs/manual/
13. jsPDF. (n.d.). jsPDF: Generate PDF Files with JavaScript. GitHub. Retrieved from https://github.om/parallax/jsPDF
14. Google Developers. (n.d.). Web Speech API Overview. Retrieved from https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API