PROJECT REPORT
DIGITAL WHITE BOARD

By

KIRUTHIKA V

Abstract

The Digital Whiteboard is an innovative tool designed to enhance interactive learning, collaboration, and creativity. This web-based application allows users to engage in real-time drawing, note-taking, and design, providing a versatile platform for education, brainstorming, and presentations.

It includes a variety of tools such as pencils, erasers, brushes, shapes, and text options to facilitate a wide range of creative activities. Additionally, the whiteboard supports customizable background colors and export options, including saving content as PDF files. With features like sticky notes, zoom functions, undo/redo, and a grid layout, the Digital Whiteboard enables a seamless user experience.

It also offers an intuitive interface with easy-to-use controls for importing, saving, and sharing work, making it an essential tool for both individuals and teams. This report discusses the design, functionality, and features of the Digital Whiteboard, highlighting its potential for diverse use cases in education, business, and creative industries.

Table of Contents

1. Introduction

The Digital Whiteboard is a modern, interactive tool designed to replicate the traditional whiteboard experience in a virtual environment. As technology continues to advance, digital tools like this have become indispensable in educational, professional, and creative settings. The Digital Whiteboard allows users to draw, write, and organize their ideas seamlessly, providing an enhanced experience for individuals. It offers a variety of tools, including brushes, shapes, colors, and text options, enabling users to express their ideas in multiple formats with precision and creativity.

This tool is particularly valuable for brainstorming sessions, lectures, workshops, presentations, and even artistic expression. The integration of features like sticky notes, zoom functionality, and PDF export options ensures users have the flexibility to save, share, and print their work effortlessly.

This report will deliver into the architecture, features, and functionalities of the Digital Whiteboard. It will examine the user interface (UI), the available drawing and editing tools, and the practical applications of the Digital Whiteboard in various fields, highlighting its role in transforming digital learning and creative workflows.

2. Project Overview

The Digital Whiteboard project is a responsive web-based application designed to provide an intuitive and interactive user experience. The front page of the application serves as the entry point, featuring a clean interface with the whiteboard name prominently displayed, accompanied by a "Login" icon and a "Go to Whiteboard" button. Clicking the " Go to Whiteboard" button directs users to the main Digital Whiteboard page. The application includes an About Section and a Feature Section, providing users with an overview of the whiteboard’s capabilities.

The application includes an About Section and a Feature Section, providing users with an overview of the whiteboard’s capabilities. The whiteboard itself integrates numerous tools, such as drawing implements, shape options, background customization, sticky notes, and export functionality. These tools are highly responsive, ensuring a seamless user experience on devices of all sizes. The interface design incorporates animation and transition effects, enhancing the overall aesthetic appeal and interactivity of the application.

Developed using HTML, CSS, and JavaScript, the Digital Whiteboard leverages modern web development practices to ensure responsiveness, accessibility, and performance. The use of animations and transitions further enriches the user experience, while the responsive design guarantees compatibility across various devices, including desktops, tablets, and smartphones.

3. Technologies Used

HTML (Hyper Text Markup Language):

HTML is the standard markup language for creating web page structure and content.
It uses tags like <h1>, <p>, and <img> to define headings, paragraphs, and images.
The <head> section contains metadata, while the <body> contains visible content.
HTML provides a semantic structure, improving accessibility and SEO.
It is the foundational layer upon which CSS and JavaScript are built.

CSS (Cascading Style Sheets):

CSS styles HTML content by defining layouts, colors, fonts, and animations.
It uses selectors like h1, .class, or #id to target specific HTML elements.
CSS allows for responsive design, adapting web pages to different screen sizes.
It can be written inline, internally in <style>, or externally in a .css file.
CSS enhances the visual appeal of websites, separating style from structure.

JAVA SCRIPT:

JavaScript adds interactivity and dynamic behavior to web pages. It can manipulate the DOM, enabling real-time content updates and animations.
JavaScript is event-driven, responding to user actions like clicks or keypresses.
It supports advanced features through APIs, such as geolocation and data fetching.
JavaScript integrates seamlessly with HTML and CSS to create engaging web experience

4. System Architecture

The system architecture of the Digital Whiteboard is based on a client-side web application.

Built using HTML and CSS, this one manages the user interface, displaying the whiteboard, tools, and navigation elements. It is responsive, ensuring the application works across various devices with smooth animations and transitions.

Powered by JavaScript, this layer handles the functionality of the whiteboard, such as drawing, shape creation, tool switching, and form validation. It ensures interactive and dynamic behavior, like displaying error messages and exporting content.

While there is no server-side data storage, this layer temporarily handles session data, like drawing content, and enables exporting files (e.g., PDFs). Future upgrades could integrate backend storage for persistent data.

This architecture ensures the Digital Whiteboard is modular, responsive, and adaptable for future updates.

5. Implementation Details

Home Page:

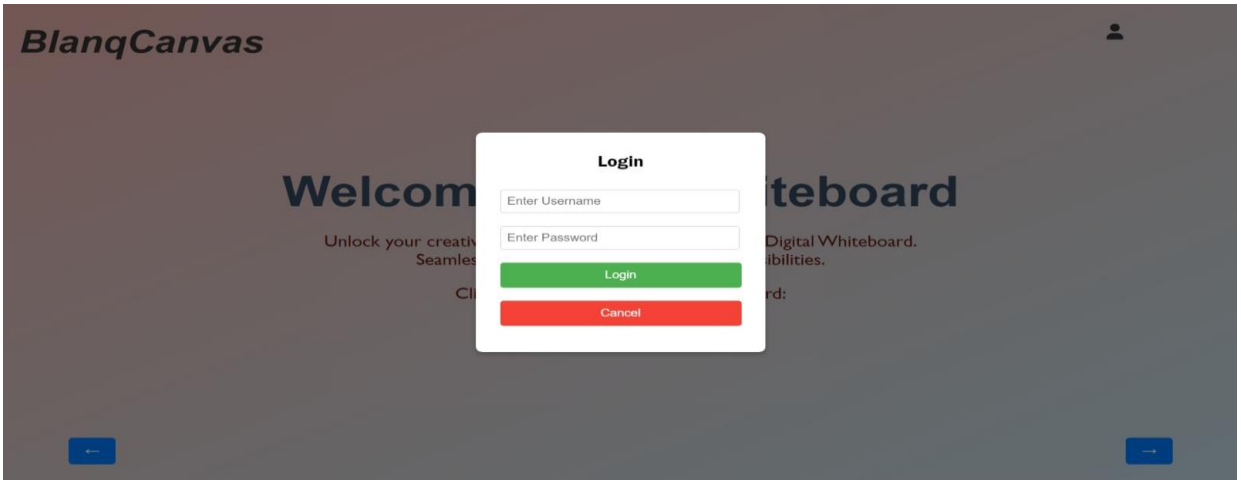
The home page serves as the landing page for the Digital Whiteboard application. It includes the whiteboard name, login icon, brief information about the whiteboard, and a button that redirects users to the main whiteboard. The page also features a slide model with previous and next buttons to navigate through multiple slides. The slide model is created using HTML for structure, CSS for styling and transitions, and JavaScript for managing the slide navigation. The "Next" and "Previous" buttons allow users to cycle through slides, ensuring smooth navigation between different sections of the home page.

Fig 1: Home page

Login:

When the user clicks on the "Login" button, a login form appears where they can input their username and password. The form includes predefined credentials (username and password) for validation. If the entered details match the predefined credentials, a pop-up message appears indicating a successful login. If the credentials are incorrect, an error message ("Invalid username or password") is displayed. If any field is left empty, the form shows a "This field is required" message. A "Cancel" button is also provided to close the form without submitting. The login functionality is implemented using HTML for the form structure, CSS for styling, and JavaScript for form validation and pop-up functionality.

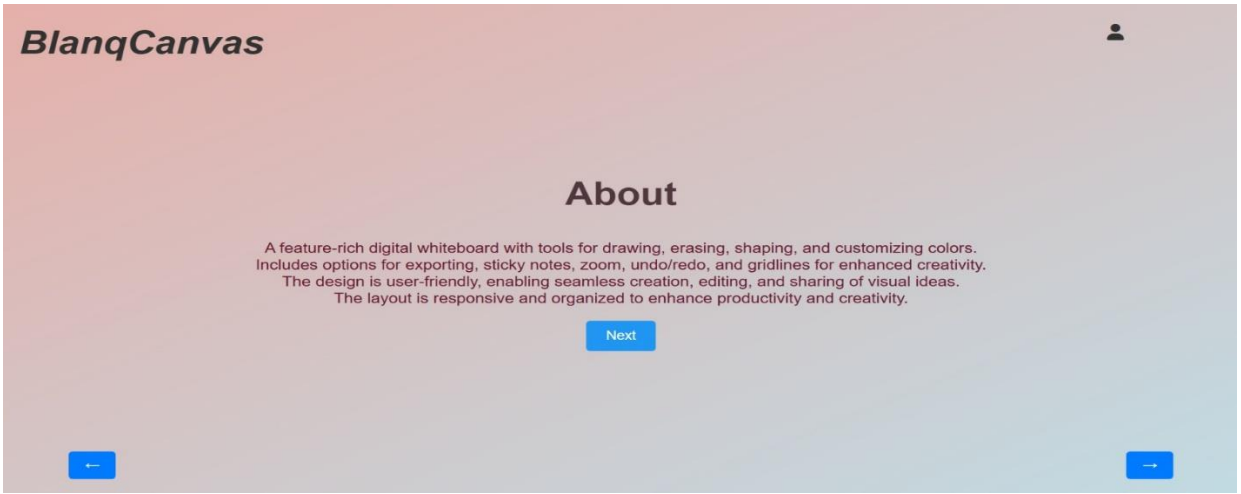
Fig 2: Login



About:

The About page provides detailed information about the Digital Whiteboard and its features. It also includes the whiteboard name, login icon, and navigation buttons (previous and next) for ease of navigation. The layout and structure of this page are designed using HTML, with CSS applied for the styling and JavaScript used for the smooth transition between slides. This page allows users to understand the tool's purpose and its practical applications.

Fig 3: About

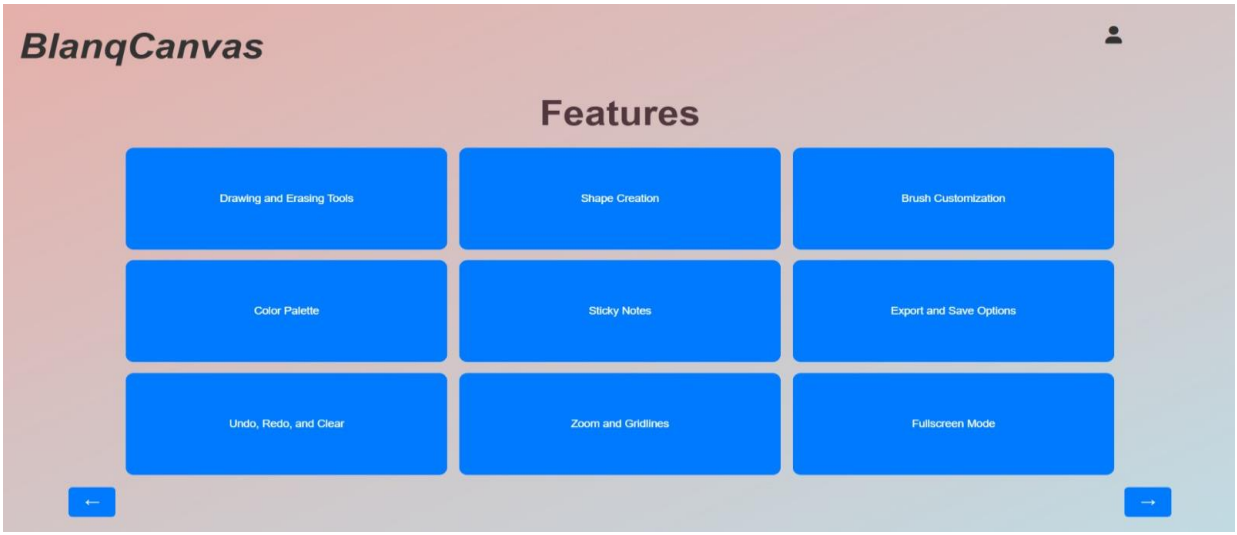


Features:

In the Feature section, the Digital Whiteboard showcases a variety of tools and features available to the user. These include:

Interactive Drawing Tools, Shape and Object Creation, Customizable Brushes, Color Picker, Sticky Notes, Export Your Work, Zoom In and Out, Gridlines and Full screen Mode, Undo and Redo Actions, Clear and Start Fresh. Each feature is clickable. When clicked, it displays a brief description or definition of that feature, helping users understand its function. Like the other pages, the whiteboard name and login icon are displayed for consistency. The feature section is implemented using HTML to structure the content, CSS for visual styling, and JavaScript to handle the interactive aspect of displaying feature descriptions.

Fig 4: Features



Digital White Board Page:

The Digital Whiteboard page is the core part of the application, where users can interact with various tools and create or edit their projects. The page includes a set of tools, a whiteboard canvas, and navigation options, allowing users to perform various actions. Here's an overview of the features and tools implemented.

Navigation and Whiteboard Setup:

Go to Home Page Button: Clicking this button redirects the user back to the home page, using simple JavaScript-based navigation.

Whiteboard: The whiteboard is a canvas element where users can draw, add shapes, or text. All the tools interact directly with this canvas to perform their respective actions.

Tools and Functionality:

The following tools have been integrated into the whiteboard for various functionalities:

New: Clears the current whiteboard, creating a blank canvas for a new project.

Import: Allows the user to import external files (such as images or documents) into the whiteboard.

Save: Saves the current project, using JavaScript to handle file-saving functionalities.

Print: Sends the current content of the whiteboard to a printer for physical output.

Share: Lets users share their projects via email or a generated link.

Zoom In / Zoom Out: Adjusts the view of the whiteboard by zooming in to show finer details or zooming out to see a broader view.

Gridlines: Adds a grid overlay to the whiteboard for better alignment and precision while working.

Full Screen: Expands the whiteboard to fill the entire screen, hiding other UI elements.

Undo / Redo: These tools allow users to revert or re-apply their most recent actions on the whiteboard.

Clear: Erases all content on the whiteboard, providing a fresh workspace.

Exit: Closes the current whiteboard project or the application.

Drawing and Shape Tools:

Pencil: A freehand drawing tool that allows users to create lines and sketches.

Eraser: Removes parts of the drawings from the canvas.

Text: Lets users add text to the whiteboard, with custom positioning.

Marker: A thicker drawing tool for bold strokes or highlighting.

Brushes: Several brush types (Crayon, Airbrush, Oil Brush, Watercolor, Calligraphy) allow users to create different artistic effects on the canvas.

Shape Tools: These tools let users draw various shapes, such as:

Square, Circle, Triangle, Ellipse, Hexagon, Star, Pentagon, Rhombus, Parallelogram, Octagon, Cross, Heart, Line, Arrow. Each tool creates its respective shape on the whiteboard, enabling precise and customizable object creation.

Colors: A color picker to choose colors for drawing, shapes, or text.

Export: Allows the user to save the whiteboard as a PDF or other formats.

Sticky Notes: Users can add sticky notes on the whiteboard for reminders or comments.

Thumbnail: Displays a small preview of the current project, often used for project navigation.

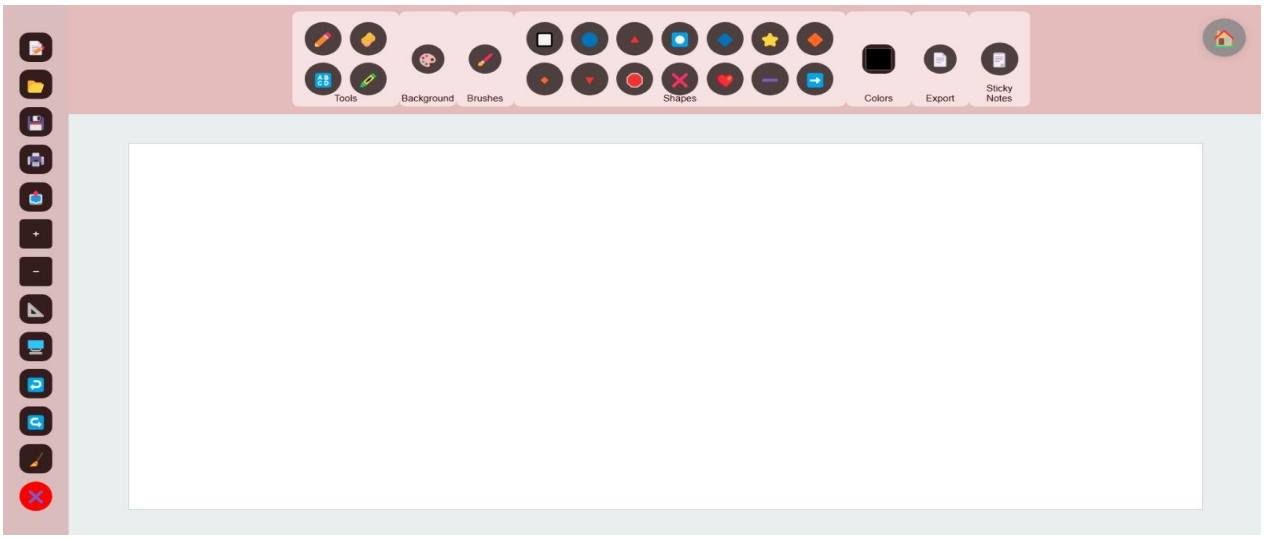
Technologies Used:

HTML: Structures the layout of the page, the whiteboard, and the tools.

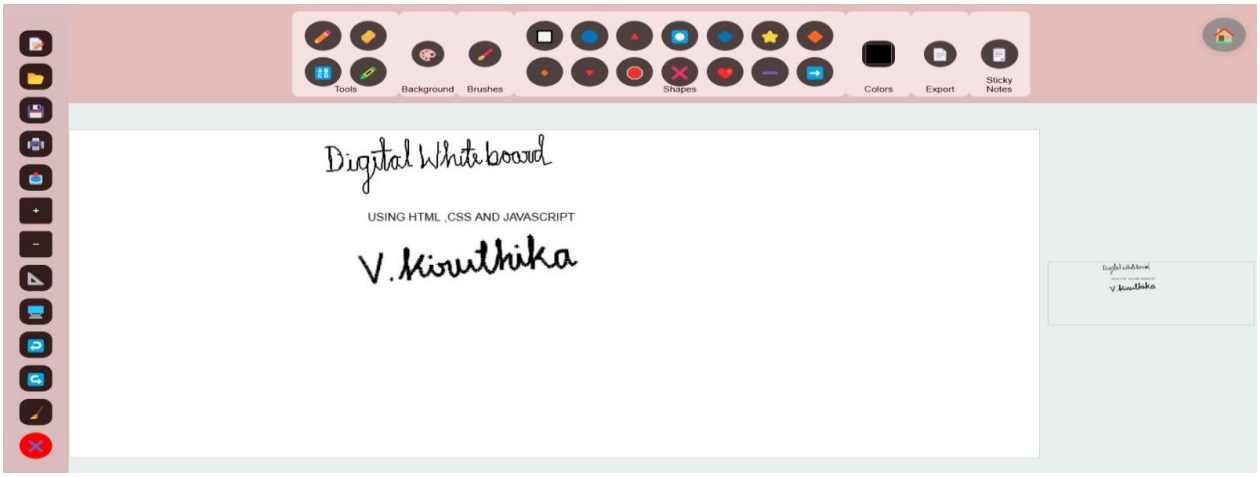
CSS: Styles the user interface and ensures the responsiveness of the whiteboard on different devices.

JavaScript: Handles the interactive behaviors of all tools, including the drawing tools, shape creation, file operations, and user actions like zooming, undo/redo, and exporting content. The responsive design ensures that all tools and functionalities are accessible across a wide range of devices.

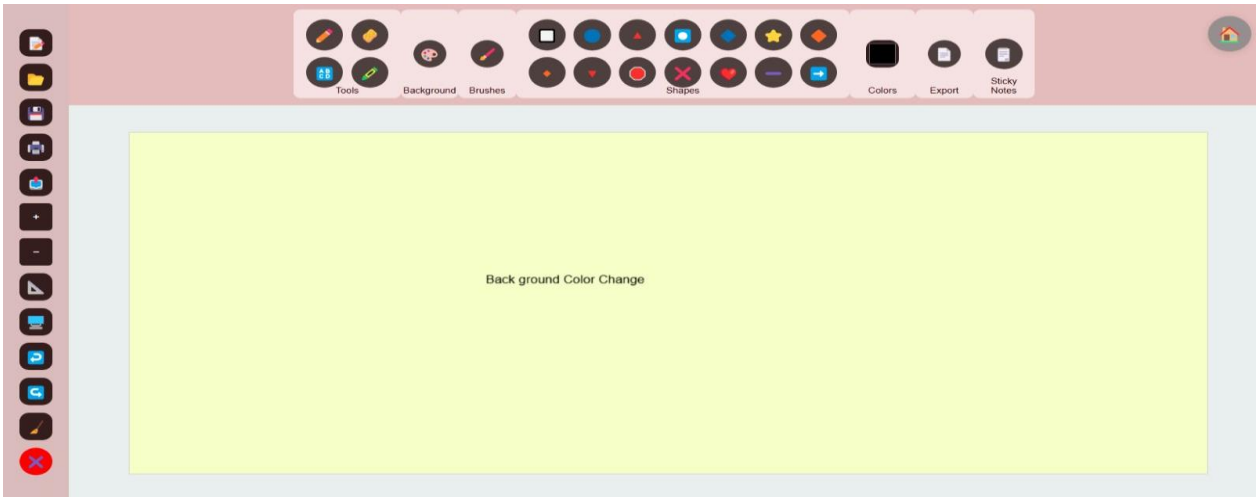
Fig 5: Digital White Board



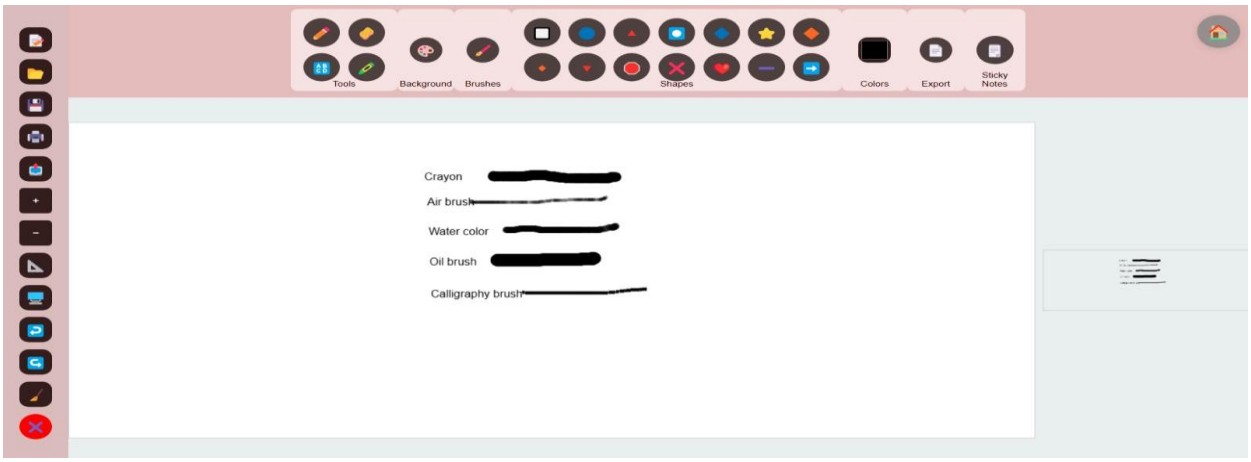
Tools:



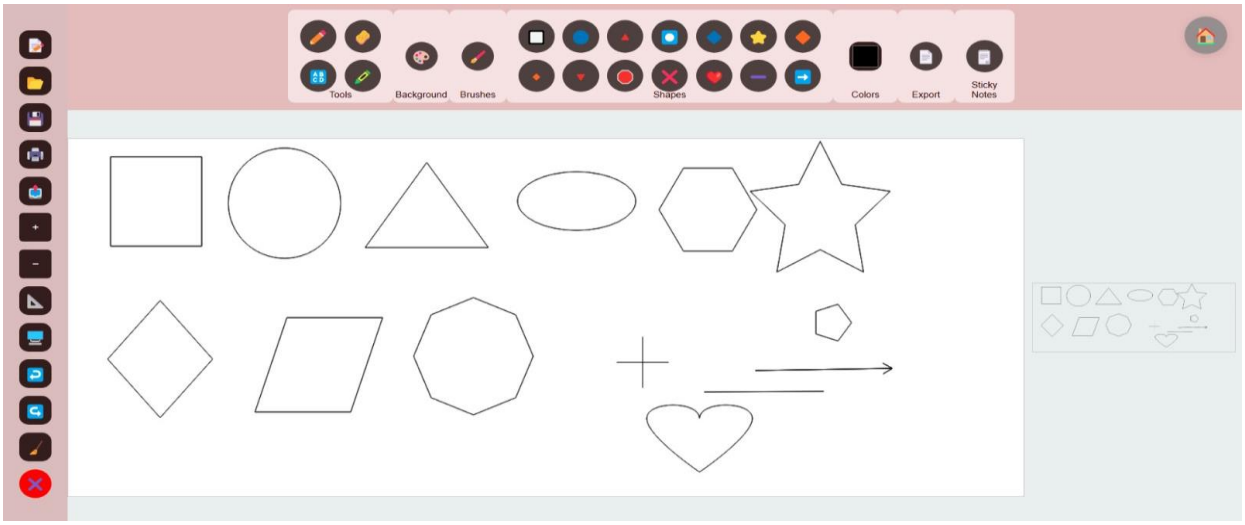
Back groud Color:



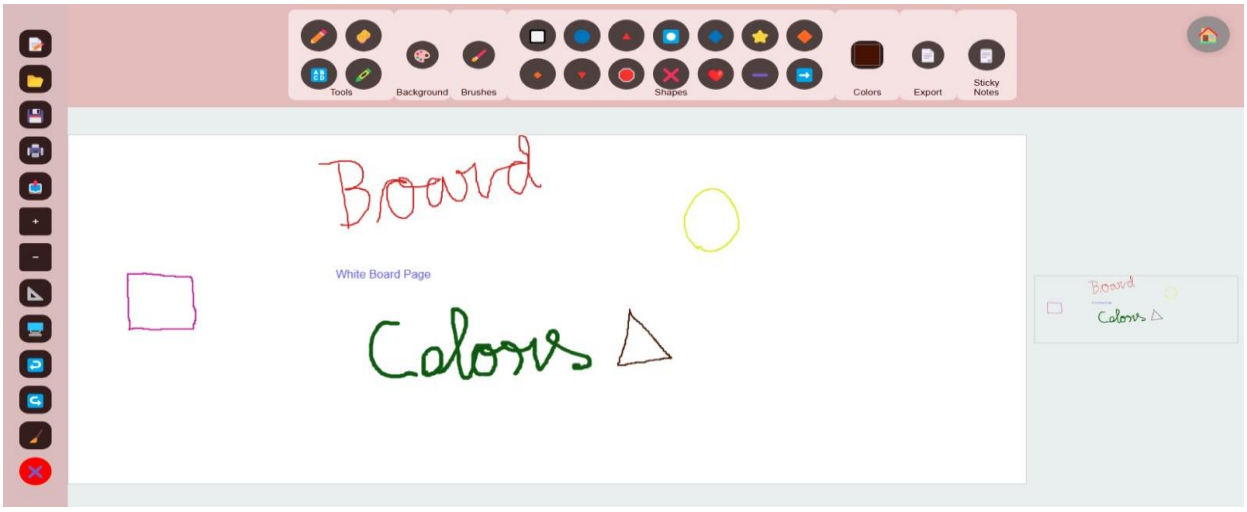
Brushes:



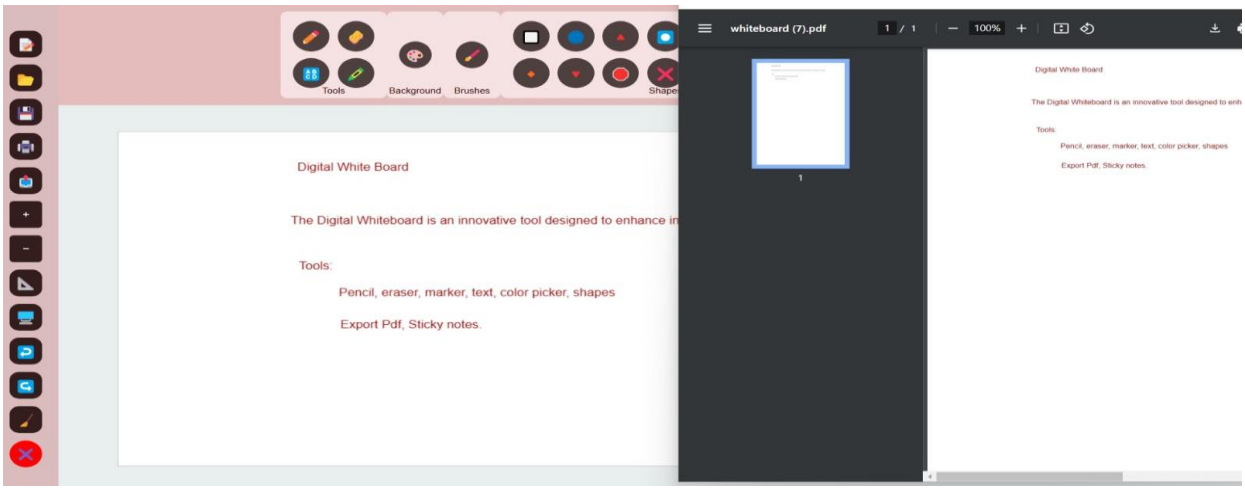
Shapes:



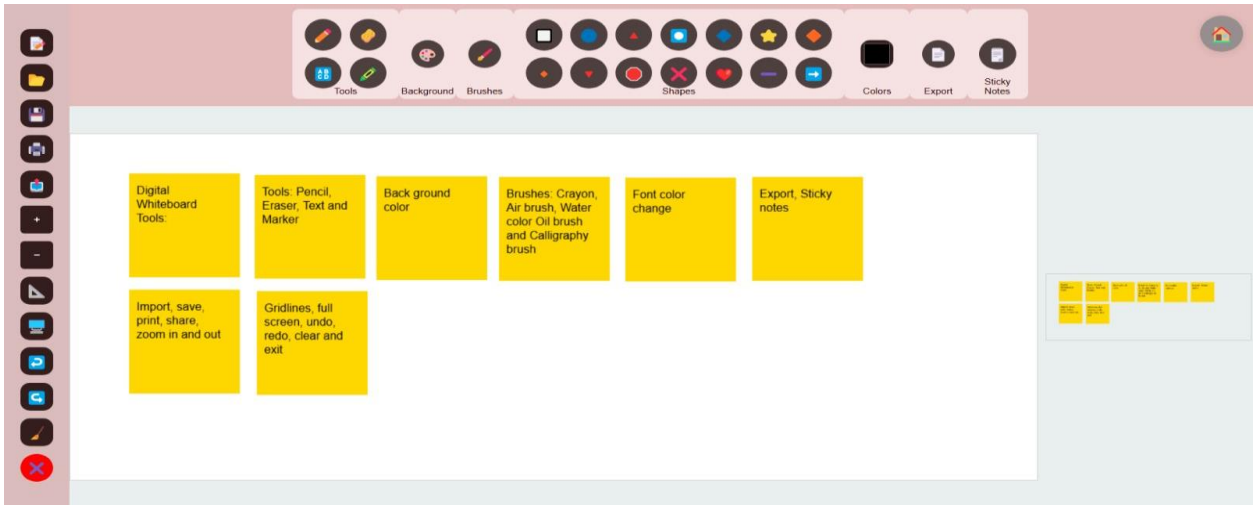
Colors:



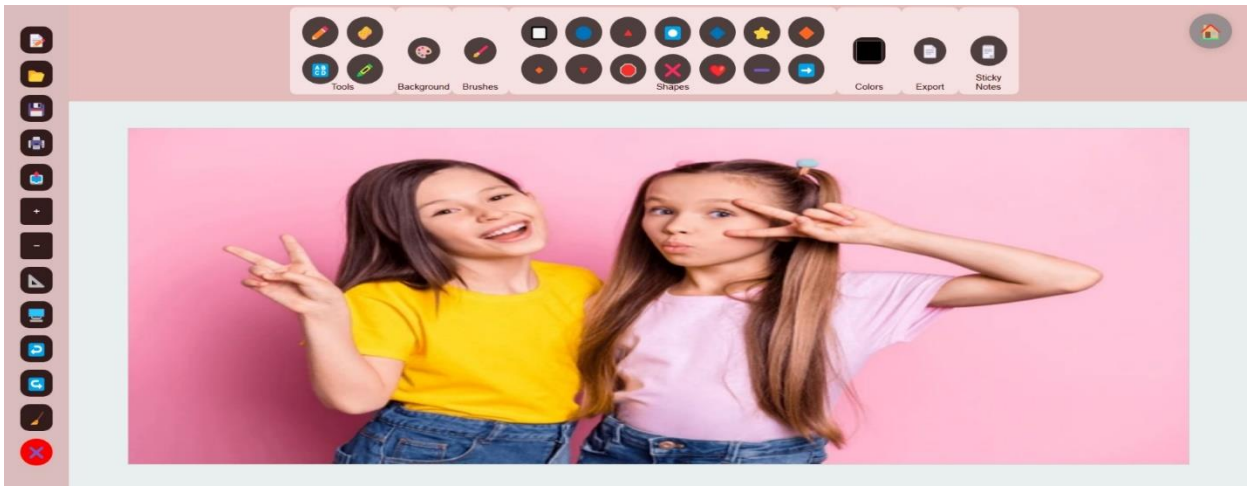
Export:



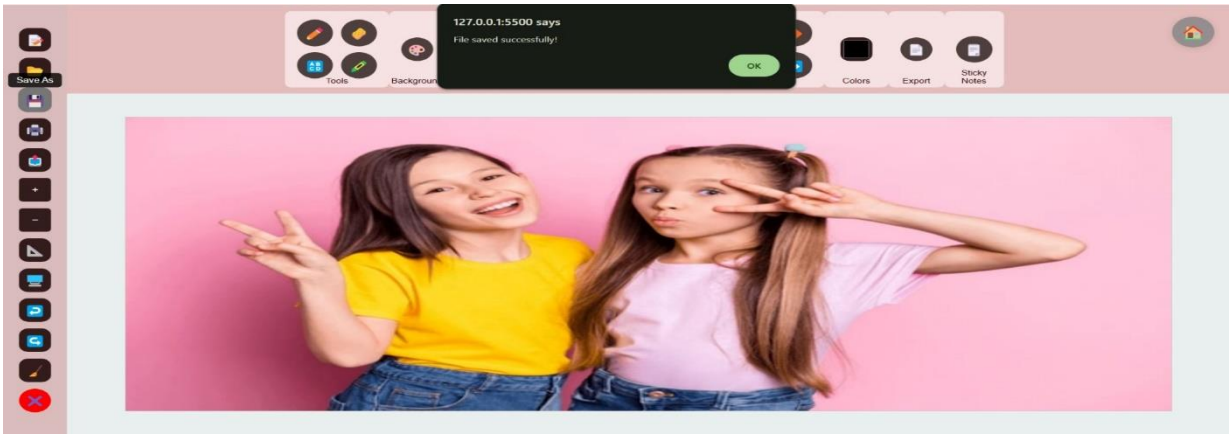
Sticky Notes:



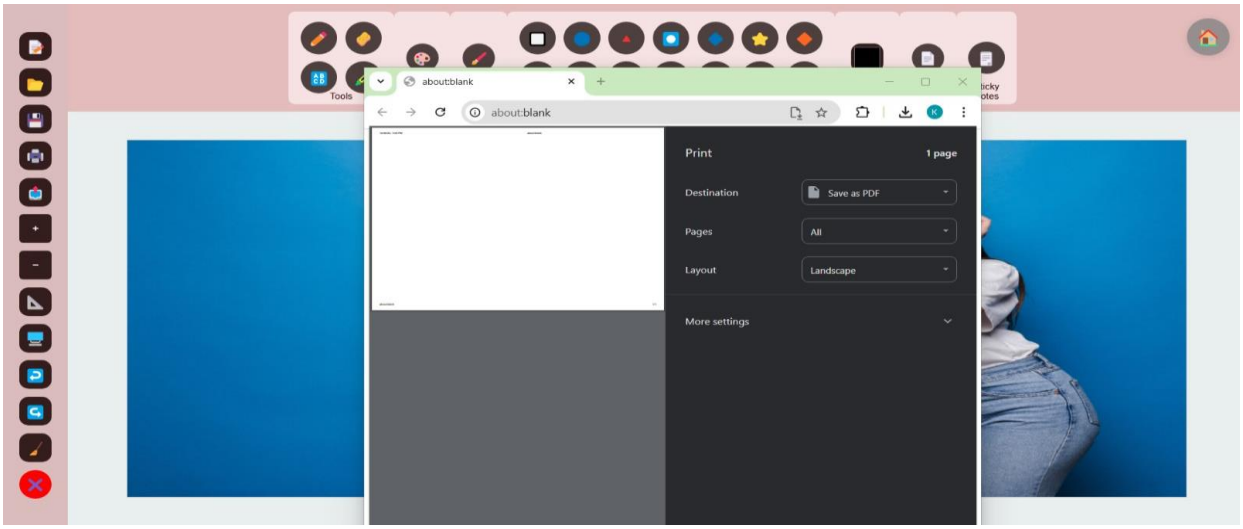
Import Image:



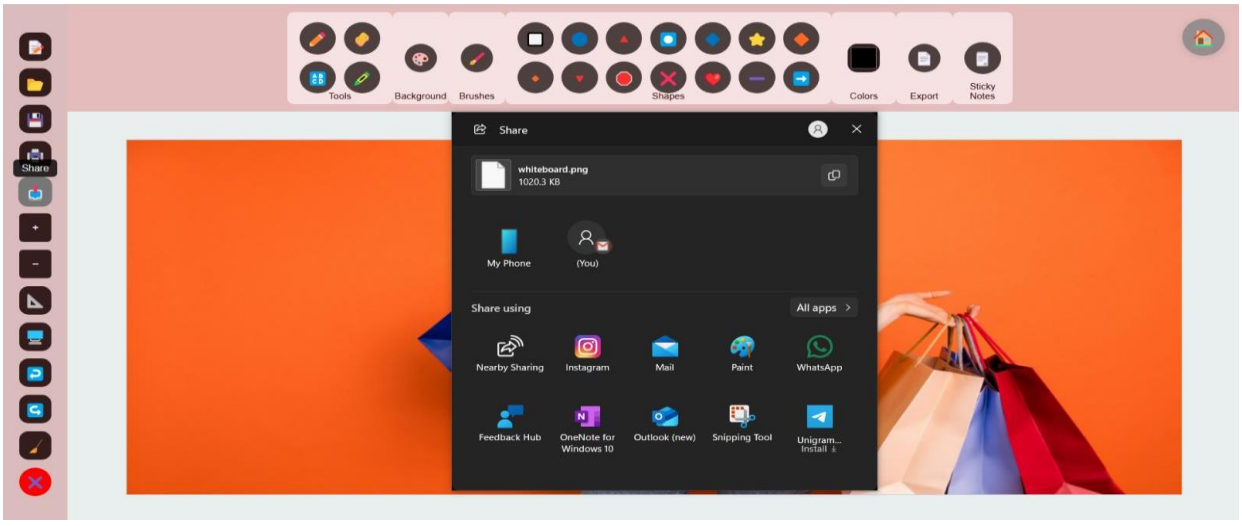
Save:



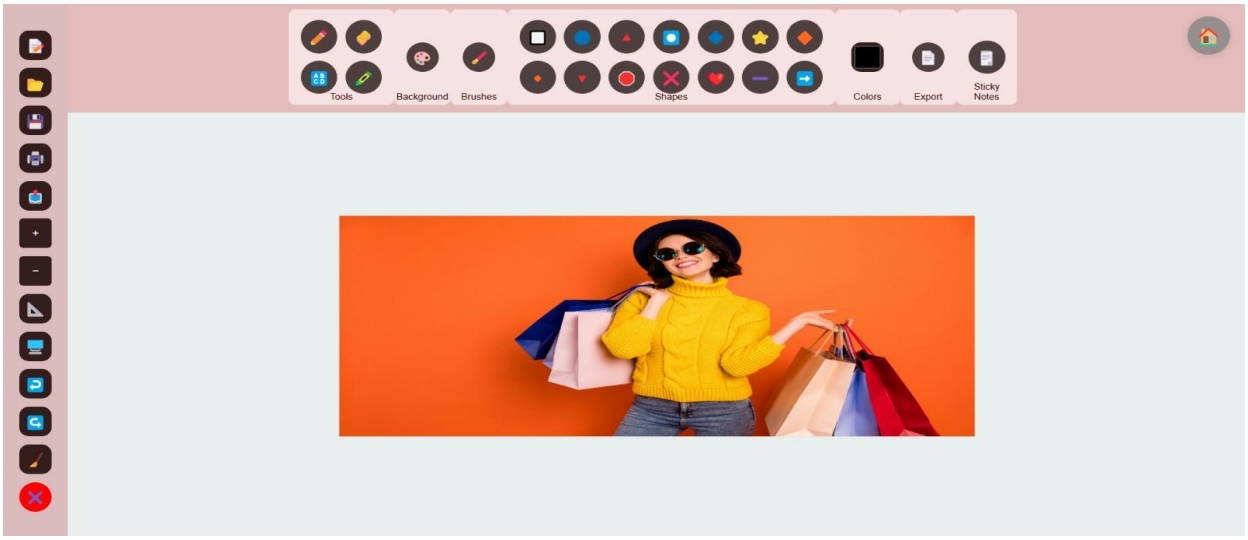
Print:



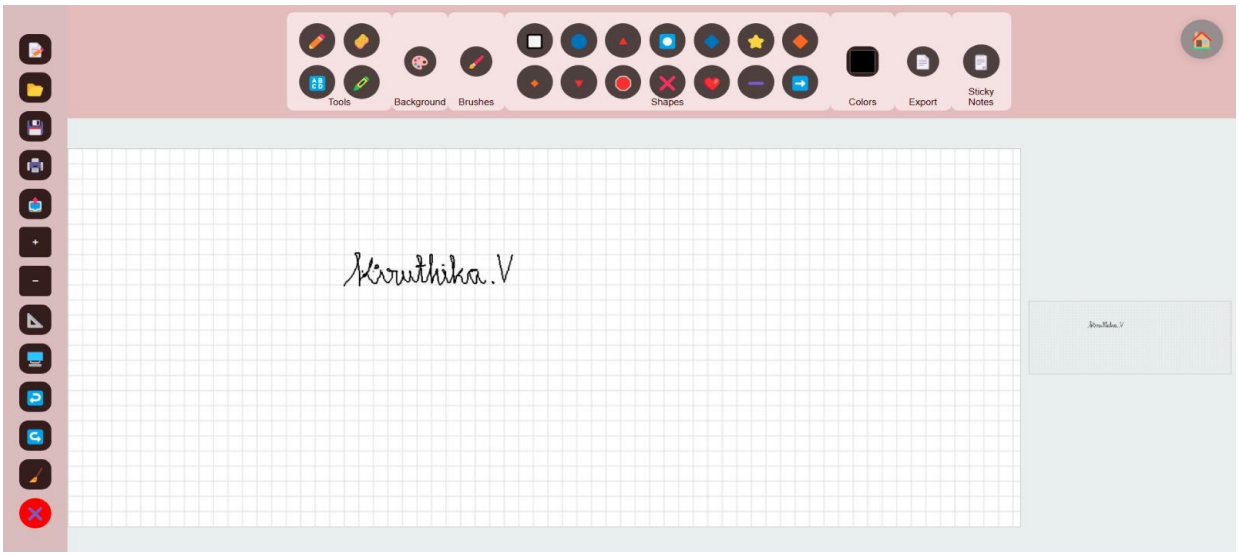
Share:



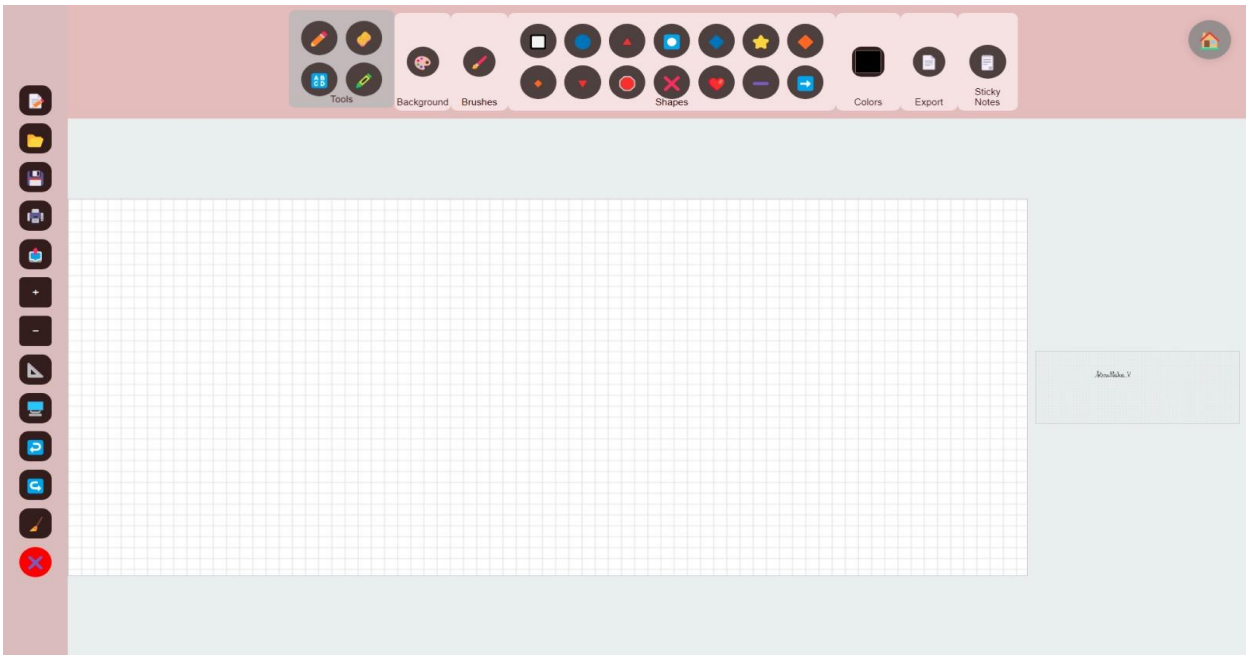
Zoom in and Zoom out:



Grid lines:



Full Screen:



Thumbnail:

Home Page:

Functionality:

The home page of the Digital Whiteboard app features the whiteboard name, login icon, brief info, and a button to access the main whiteboard. It includes a slide model with navigation buttons for smooth transitions between multiple slides. The slide model is built using HTML, CSS, and JavaScript for structure, styling, and navigation.

Technologies Used:

Front end: HTML, CSS, Java script.

Code:

Index.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Front Page</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="slider">
        <div class="slide" id="slide1">
            <div class="header">
                <span class="name"> BlanqCanvas </span>
                <i class="fas fa-user login-icon" onclick="openLoginPopup()"></i>
            </div>
            <div class="welcome-message">
                <h1>Welcome to the Whiteboard</h1>
                <p>Unlock your creativity with the dynamic features of our Digital Whiteboard.
                    <br> Seamless, intuitive, and built for limitless possibilities.</p>
                <p>Click below to start using the whiteboard:</p>
                <button><a href="whiteboard.html" class="start-button">Go to Whiteboard</a></button>
            </div>
            <div class="navigation">
                <button class="prev" onclick="goToSlide(3)">&#8592; </button>
                <button class="next" onclick="goToSlide(2)">&#8594; </button>
            </div>
        </div>
    <!-- Overlay -->
    <div class="overlay" id="overlay" onclick="closeLoginPopup()"></div>
    <!-- Login Popup -->
    <div class="login-popup" id="loginPopup">
        <h2>Login</h2>
        <input type="text" id="username" placeholder="Enter Username">
        <input type="password" id="password" placeholder="Enter Password">
        <button onclick="login()">Login</button>
        <button class="cancel-button" onclick="closeLoginPopup()">Cancel</button>
```

```
    </div>
        <div class="slide" id="slide2">
            <div class="header">
                <span class="name"> BlanqCanvas </span>
                <i class="fas fa-user login-icon" onclick="openLoginPopup()"></i>
            </div>
            <h2>About</h2>
            <p class="definition">A feature-rich digital whiteboard with tools for drawing, erasing, shaping, and customizing colors.<br>
                Includes options for exporting, sticky notes, zoom, undo/redo, and gridlines for enhanced creativity. <br>
                The design is user-friendly, enabling seamless creation, editing, and sharing of visual ideas. <br>
                The layout is responsive and organized to enhance productivity and creativity.
            </p>
            <button onclick="goToSlide(3)">Next</button>
            <div class="navigation">
                <button class="prev" onclick="goToSlide(1)">&#8592; </button>
                <button class="next" onclick="goToSlide(3)">&#8594; </button>
            </div>
        </div>
        <div class="slide" id="slide3">
            <div class="header">
                <span class="name"> BlanqCanvas </span>
                <i class="fas fa-user login-icon" onclick="openLoginPopup()"></i>
            </div>
            <h2>Features</h2>
            <div class="features">
                <button class="feature-btn" onclick="showFeature(this, 'Drawing and Erasing Tools')">
                    <div class="front">Drawing and Erasing Tools</div>
                    <div class="back">Intuitive tools like pencil, marker, and eraser for effortless sketching and editing.
                        </div>
                </button>
                <button class="feature-btn" onclick="showFeature(this, 'Shape Creation')">
                    <div class="front">Shape Creation</div>
                    <div class="back">Predefined shapes such as squares, circles, arrows, and stars for structured diagrams.
```

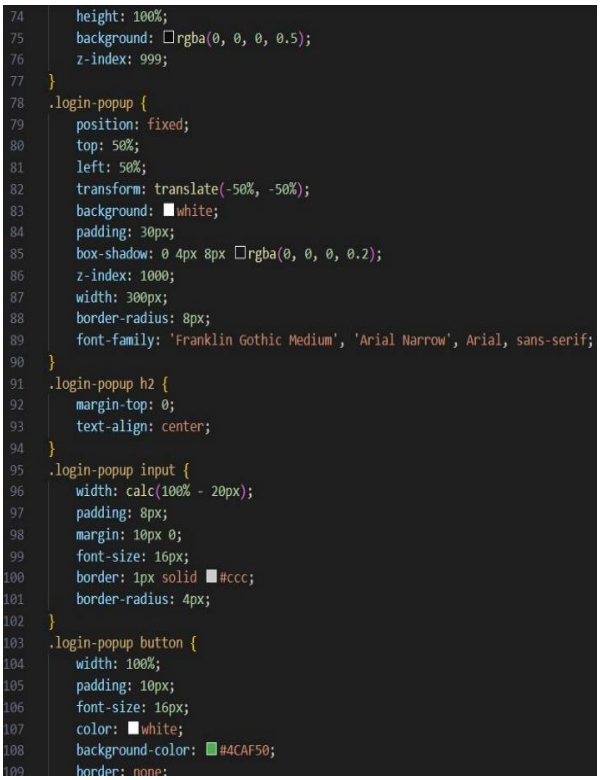
```
                    </div>
                </button>
                <button class="feature-btn" onclick="showFeature(this, 'Brush Customization')">
                    <div class="front">Brush Customization</div>
                    <div class="back">Multiple brush styles, including watercolor, crayon, and calligraphy options.</div>
                </button>
                <button class="feature-btn" onclick="showFeature(this, 'Color Palette')">
                    <div class="front">Color Palette</div>
                    <div class="back">A color picker for selecting and applying your preferred brush or background color.</div>
                </button>
                <button class="feature-btn" onclick="showFeature(this, 'Sticky Notes')">
                    <div class="front">Sticky Notes</div>
                    <div class="back">Add and manage sticky notes for brainstorming or jotting down quick ideas.
                    </div>
                </button>
                <button class="feature-btn" onclick="showFeature(this, 'Export and Save Options')">
                    <div class="front">Export and Save Options</div>
                    <div class="back">Export your work as a PDF or save files for future use.
                    </div>
                </button>
                <button class="feature-btn" onclick="showFeature(this, 'Undo, Redo, and Clear')">
                    <div class="front">Undo, Redo, and Clear</div>
                    <div class="back">Simple controls to fix mistakes or clear the canvas for a fresh start.</div>
                </button>
                <button class="feature-btn" onclick="showFeature(this, 'Zoom and Gridlines')">
                    <div class="front">Zoom and Gridlines</div>
                    <div class="back">Zoom in/out and toggle gridlines for precise alignment and focus.</div>
                </button>
                <button class="feature-btn" onclick="showFeature(this, 'Fullscreen Mode')">
                    <div class="front">Fullscreen Mode</div>
                    <div class="back">Switch to fullscreen for an immersive whiteboard experience.
                    </div>
```

```
                    </button>
                </div>
                <div class="navigation">
                    <button class="prev" onclick="goToSlide(2)">&#8592; </button>
                    <button class="next" onclick="goToSlide(1)">&#8594; </button>
                </div>
            </div>
        <script src="script.js"></script>
    </body>
    </html>
```

Styles.css:

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background-color: #f4f4f9;
}
/* Slider styles */
.slider {
    width: 100%;
    height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
    background-color: #fadbd8 ;
    background-image: linear-gradient(to bottom right , #e6b0aa , #c1dbe3 );
    background-size:contain;
    background-position: center;
    position: relative;
}
/* Slide styles */
.slide {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    text-align: center;
    color: #b67c7c;
    width: 100%;
```

```css
    height: 100%;
    animation: fadeIn 1s ease-in-out;
}
/* Header section */
.header {
    position: absolute;
    top: 20px;
    left: 20px;
    display: flex;
    justify-content: space-between;
    width: 90%;
    padding: 10px 0;
}
.name {
    font-size: 3rem;
    font-weight: bold;
    margin-right: 10px;
    color: #333;
    font-style:italic;
    animation: slideInLeft 1s ease-in-out;
}
/* Login */
.login-icon {
    font-size: 24px;
    color: #333;
    width: 40px;
    cursor: pointer;
}
.login-popup, .overlay {
    display: none;
}
.overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
```



```css
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
    z-index: 999;
}
.login-popup {
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background: white;
    padding: 30px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    z-index: 1000;
    width: 300px;
    border-radius: 8px;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
.login-popup h2 {
    margin-top: 0;
    text-align: center;
}
.login-popup input {
    width: calc(100% - 20px);
    padding: 8px;
    margin: 10px 0;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 4px;
}
.login-popup button {
    width: 100%;
    padding: 10px;
    font-size: 16px;
    color: white;
    background-color: #4CAF50;
    border: none;
```

```css
    margin: 10px 0;
    border-radius: 4px;
    cursor: pointer;
}
.login-popup button:hover {
    background-color: #2b5c2e;
}
.login-popup .cancel-button {
    background-color: #f44336;
}
.login-popup .cancel-button:hover {
    background-color: #620903;
}
/* Welcome message */
.welcome-message h1 {
    font-size: 4rem;
    margin-bottom: 20px;
    color: #34495e ;
    animation: fadeInUp 1.5s ease-out;
}
.welcome-message p {
    font-size: 1.5rem;
    font-family:'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif, Cochin, Georgia, Times, 'Times New Roman', serif;
    margin-bottom: 20px;
    color: #641e16 ;
    animation: fadeInUp 1.5s ease-out;
}
button {
    padding: 12px 25px;
    font-size: 18px;
    background-color: #2196F3;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    margin-top: 10px;
```

```
    transition: background-color 0.3s ease, transform 0.3s ease;
}
button a{
    text-decoration: none;
}
button:hover {
    background-color: #1b8c28;
    transform: scale(1.1);
}
.prev, .next {
    padding: 10px 20px;
    font-size: 18px;
    background-color: rgba(0, 0, 0, 0.5);
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.3s ease;
}
.prev:hover, .next:hover {
    background-color: rgba(66, 59, 59, 0.7);
    transform: scale(1.1);
}
#slide1 {
    display: flex;
    flex-direction: column;
}
#slide2 h2, #slide3 h2 {
    font-size: 3rem;
    margin-bottom: 20px;
    color: #51383f;
}
#slide2 p{
    font-size: 1.3rem;
    color: #68182f;
}
```

```
/* Features */
.features {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 15px;

}
.feature-btn {
    position: relative;
    width: 400px;
    padding: 70px;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 10px;
    cursor: pointer;
    transform: translateY(0);
    transition: transform 0.5s ease, background-color 0.3s ease;
    perspective: 1000px;
    font-size: 16px;
}
.feature-btn:hover {
    transform: translateY(-10px);
    background-color: #0056b3;
}
.feature-btn.active .back {
    transform: rotateY(0);
}
.feature-btn .front, .feature-btn .back {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    backface-visibility: hidden;
    border-radius: 10px;
    display: flex;
```

```
        justify-content: center;
        align-items: center;
        font-size: 14px;
    }
    .feature-btn .front{
        animation: slideInLeft 1s ease-in-out;
    }
    .feature-btn .back {
        background-color: #6f6565;
        color: #f4eaea;
        transform: rotateY(180deg);
    }
    .feature-btn.active .front {
        transform: rotateY(180deg);
    }
    .feature-btn.active .back {
        transform: rotateY(0);
    }
    /* Navigation buttons */
    .navigation {
        position: absolute;
        bottom: 20px;
        display: flex;
        justify-content: space-between;
        width: 90%;
        animation: fadeInUp 1s ease-in-out;
    }

    .navigation button {
        padding: 10px 20px;
        background-color: #007bff;
        color: #fff;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        margin: 5px;
```


```
/* Media queries for responsiveness */
@media (max-width: 1250px) {
    .feature-btn {
        width: 300px;
        padding: 50px;
    }
    .features {
        display: grid;
        grid-template-columns: repeat(3, 1fr);
        gap: 15px;
    }
    .navigation button {
        font-size: 14px;
        padding: 8px 16px;
    }
    .name {
        font-size: 3rem;
    }
}
@media (max-width: 960px) {
    .header {
        flex-direction: column;
        align-items: flex-start;
    }
    .login-icon {
        width: 30px;
    }
    .name {
        font-size: 2rem;
    }
    button {
        font-size: 14px;
        padding: 8px 16px;
    }
    .prev,
    .next {
        font-size: 14px;
```

```
        padding: 8px ;
    }
    .feature-btn {
        width: 250px;
        padding: 60px;
    }
    .features {
        display: grid;
        grid-template-columns: repeat(2, 1fr);
        gap: 10px;
    }
}
@media (max-width: 540px) {
    .welcome-message h1 {
        font-size: 24px;
    }
    button {
        font-size: 12px;
        padding: 6px 12px;
    }
    .feature-btn {
        width: 150px;
        padding: 50px;
    }
    .features {
        display: grid;
        grid-template-columns: repeat(2, 1fr);
        gap: 15px;
    }
    .prev,
    .next {
        font-size: 10px;
        padding: 3px ;
    }
}
```

Script.js:

JavaScript code for a slide model with previous and next buttons to navigate through multiple slides along with login form validation and pop-up functionality.

```
let currentSlide = 1;
function goToSlide(slideNumber) {
    const slides = document.querySelectorAll('.slide');
    slides.forEach((slide, index) => {
        if (index === slideNumber - 1) {
            slide.style.display = 'flex';
        } else {
            slide.style.display = 'none';
        }
    });
    currentSlide = slideNumber;
}
function showFeature(feature) {
    const featureInfo = document.getElementById('feature-info');
    if (feature === 'feature1') {
        featureInfo.innerHTML = "<p>Feature 1: This is the first feature description.</p>";
    } else if (feature === 'feature2') {
        featureInfo.innerHTML = "<p>Feature 2: This is the second feature description.</p>";
    } else if (feature === 'feature3') {
        featureInfo.innerHTML = "<p>Feature 3: This is the third feature description.</p>";
    }
}
function showFeature(button, featureText) {
    const allButtons = document.querySelectorAll('.feature-btn');
    allButtons.forEach(btn => btn.classList.remove('active'));
    button.classList.add('active');
}
function openLoginPopup() {
    document.getElementById("loginPopup").style.display = "block";
    document.getElementById("overlay").style.display = "block";
}
function closeLoginPopup() {
    document.getElementById("loginPopup").style.display = "none";
    document.getElementById("overlay").style.display = "none";
}
function login() {
    const username = document.getElementById("username").value;
```

```
    const password = document.getElementById("password").value;
    if (username === "admin" && password === "12345") {
        alert("Login Successful!");
        closeLoginPopup();
    } else {
        alert("Invalid Username or Password");
    }
}
// Initialize the first slide
goToSlide(1);
```

Whiteboard Page:

Functionality:

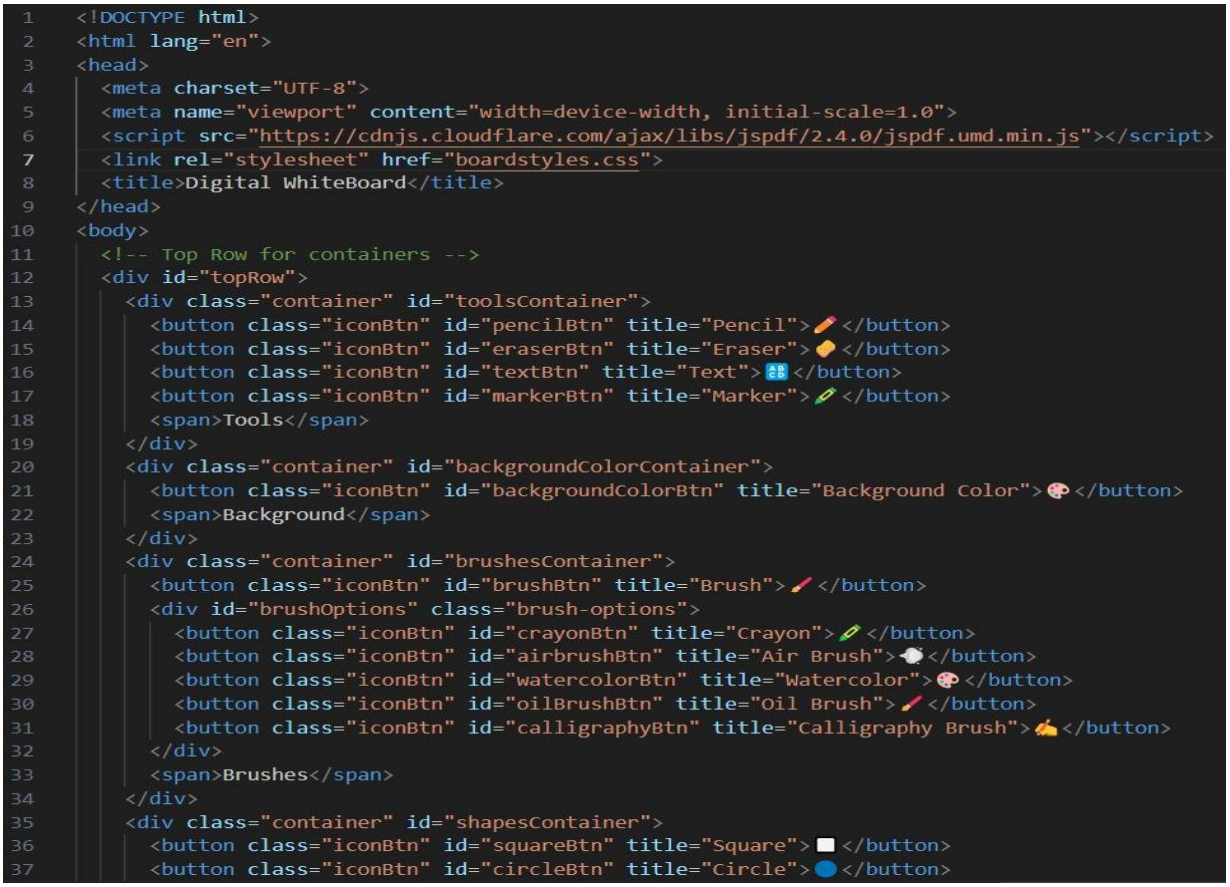
The Digital Whiteboard page is the core part of the application, where users can interact with various tools and create or edit their projects. The page includes a set of tools, a whiteboard canvas, and navigation options, allowing users to perform various actions. Here's an overview of the features and tools implemented.

Technologies Used:

Front end: HTML, CSS, Javascript.

Code:

Whiteboard.html:



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.4.0/jspdf.umd.min.js"></script>
  <link rel="stylesheet" href="boardstyles.css">
  <title>Digital WhiteBoard</title>
</head>
<body>
  <!-- Top Row for containers -->
  <div id="topRow">
    <div class="container" id="toolsContainer">
      <button class="iconBtn" id="pencilBtn" title="Pencil"> </button>
      <button class="iconBtn" id="eraserBtn" title="Eraser"> </button>
      <button class="iconBtn" id="textBtn" title="Text"> </button>
      <button class="iconBtn" id="markerBtn" title="Marker"> </button>
      <span>Tools</span>
    </div>
    <div class="container" id="backgroundColorContainer">
      <button class="iconBtn" id="backgroundColorBtn" title="Background Color"> </button>
      <span>Background</span>
    </div>
    <div class="container" id="brushesContainer">
      <button class="iconBtn" id="brushBtn" title="Brush"> </button>
      <div id="brushOptions" class="brush-options">
        <button class="iconBtn" id="crayonBtn" title="Crayon"> </button>
        <button class="iconBtn" id="airbrushBtn" title="Air Brush"> </button>
        <button class="iconBtn" id="watercolorBtn" title="Watercolor"> </button>
        <button class="iconBtn" id="oilBrushBtn" title="Oil Brush"> </button>
        <button class="iconBtn" id="calligraphyBtn" title="Calligraphy Brush"> </button>
      </div>
      <span>Brushes</span>
    </div>
    <div class="container" id="shapesContainer">
      <button class="iconBtn" id="squareBtn" title="Square"> </button>
      <button class="iconBtn" id="circleBtn" title="Circle"> </button>
```

```
      <button class="iconBtn" id="triangleBtn" title="Triangle">🔺</button>
      <button class="iconBtn" id="ellipseBtn" title="Ellipse">🔵</button>
      <button class="iconBtn" id="hexagonBtn" title="Hexagon">🔷</button>
      <button class="iconBtn" id="starBtn" title="Star">⭐</button>
      <button class="iconBtn" id="pentagonBtn" title="Pentagon">🔶</button>
      <button class="iconBtn" id="rhombusBtn" title="Rhombus">🔸</button>
      <button class="iconBtn" id="parallelogramBtn" title="Parallelogram">🔻</button>
      <button class="iconBtn" id="octagonBtn" title="Octagon">🛑</button>
      <button class="iconBtn" id="crossBtn" title="Cross">❌</button>
      <button class="iconBtn" id="heartBtn" title="Heart">❤️</button>
      <button class="iconBtn" id="lineBtn" title="Line">➖</button>
      <button class="iconBtn" id="arrowBtn" title="Arrow">➡️</button>
      <span>Shapes</span>
    </div>
    <div class="container" id="colorsContainer">
      <input class="tool" type="color" id="color-picker" title="Pick a color" onchange="changeBrushColor(this.value)">
      <span>Colors</span>
    </div>
    <div class="container" id="exportContainer">
      <button class="iconBtn" id="exportPdfBtn" title="Export PDF">📄</button>
      <span>Export</span>
    </div>
    <div id="stickyNotesContainer" class="container">
      <button class="iconBtn" id="addStickyNoteBtn" title="Add Sticky Note">📝</button>
      <span>Sticky Notes</span>
    </div>
  </div>
  <!-- Left Sidebar for tools -->
  <div id="toolbar">
    <div class="tool-container">
      <button class="tool" data-name="New" id="newBtn">📝</button>
    </div>
    <div class="tool-container">
```

```
      <button class="tool" data-name="Import" id="importBtn">📂</button>
    </div>
    <div class="tool-container">
      <button class="tool" data-name="Save As" id="saveBtn">💾</button>
    </div>
    <div class="tool-container">
      <button class="tool" data-name="Print" id="printBtn">🖨️</button>
    </div>
    <div class="tool-container">
      <button class="tool" data-name="Share" id="shareBtn">📩</button>
    </div>
    <div class="tool-container">
      <button class="tool zoom-btn" data-name="Zoom In" id="zoomInBtn">+</button>
    </div>
    <div class="tool-container">
      <button class="tool zoom-btn" data-name="Zoom Out" id="zoomOutBtn">-</button>
    </div>
    <div class="tool-container">
      <button class="tool" data-name="Gridlines" id="gridBtn">📐</button>
    </div>
    <div class="tool-container">
      <button class="tool" data-name="Full Screen" id="fullscreenBtn">🖥️</button>
    </div>
    <div class="tool-container">
      <button class="tool" data-name="Undo" id="undoBtn">↩️</button>
    </div>
    <div class="tool-container">
      <button class="tool" data-name="Redo" id="redoBtn">↪️</button>
    </div>
    <div class="tool-container">
      <button class="tool" data-name="Clear" id="clearBtn">🧹</button>
    </div>
    <div class="tool-container">
```

```
      <button class="tool exit-btn" data-name="Exit" id="exitBtn">❌</button>
    </div>
  </div>
  <!-- Go to Home Button -->
  <a href="index.html" id="goHomeBtn" title="Go to Home">🏠</a>
  <!-- Main Content Area -->
  <div id="content">
    <!-- Whiteboard -->
    <div id="whiteboard-wrapper">
      <canvas id="whiteboard"></canvas>
    </div>
    <!-- Thumbnail -->
    <div>
      <img id="thumbnailPreview" alt="Thumbnail Preview" style="display: none; margin: 10px; border: 1px solid #ccc;" />
    </div>
  </div>
  <script src="boardscript.js"></script>
</body>
</html>
```

Boardstyle.css:

```
body {
  margin: 0;
  display: flex;
  flex-direction: column;
  height: 100vh;
  background-color: #e9eeee;
  font-family: Arial, sans-serif;
}
/* Top Row */
#topRow {
  display: flex;
  justify-content: center;
  padding: 10px;
  background-color: #e5bcbc;
  margin-left: 70px;
  flex-direction: row;
}
/* Main Container */
.container {
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: center;
  gap: 20px;
  background-color: #f6e2e2;
  border-radius: 8px;
  padding: 15px;
  color: rgb(242, 86, 86);
  position: relative;
  transition: background-color 0.3s ease, transform 0.3s ease;
}
.container:hover {
  background-color: #aca9a9;
  transform: translateY(-5px);
}
.container span {
  position: absolute;
  bottom: 5px;
  left: 50%;
  transform: translateX(-50%);
  color: rgb(43, 2, 2);
  font-size: 12px;
  padding: 0 5px;
  transition: color 0.3s ease;
}
/* Icon Button */
.iconBtn {
  background-color: #4f4040;
  border: none;
  color: white;
  font-size: 20px;
  width: 40px;
  height: 40px;
  display: flex;
  align-items: center;
  justify-content: center;
  border-radius: 50%;
  cursor: pointer;
  position: relative;
  transition: transform 0.3s ease, background-color 0.3s ease;
}
.iconBtn:hover {
  background-color: #777;
  border-radius: 5px;
  transform: scale(1.1);
}
.iconBtn::after {
  content: attr(title);
  position: absolute;
  bottom: 100%;
  left: 50%;
  transform: translateX(-50%);
  background-color: rgba(0, 0, 0, 0.8);
```

```
  color: #fff;
  font-size: 12px;
  padding: 5px 10px;
  border-radius: 5px;
  white-space: nowrap;
  max-width: 120px;
  text-align: center;
  visibility: hidden;
  opacity: 0;
  transition: visibility 0.2s, opacity 0.2s;
  z-index: 10;
}
.iconBtn:hover::after {
  visibility: visible;
  opacity: 1;
}
/* Tool containers */
#toolsContainer, #shapesContainer, #colorsContainer, #stickyNotesContainer {
  display: flex;
  flex-direction: row;
  gap: 5px;
  justify-content: center;
}
#toolsContainer .iconBtn,
#shapesContainer .iconBtn,
#colorsContainer .iconBtn,
#stickyNotesContainer .iconBtn {
  width: 45px;
  height: 45px;
}
#backgroundColorContainer,
#importContainer {
  display: flex;
  align-items: center;
}
#backgroundColorBtn {
```

```
    font-size: 20px;
    padding: 5px;
  }
  #importPdfBtn {
    font-size: 20px;
    padding: 5px;
  }
  /* Whiteboard Section */
  #content {
    display: flex;
    justify-content: center;
    align-items: center;
    flex: 1;
    margin-left: 80px;
  }
  #whiteboard-wrapper {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 80%;
    width: 90%;
  }
  #whiteboard {
    width: 100%;
    height: 500px;
    background-color: #fff;
    cursor: crosshair;
    border: 1px solid #ccc;
    box-sizing: border-box;
  }
  /* Go Home Button */
  #goHomeBtn {
    position: fixed;
    top: 20px;
    right: 20px;
    background-color: #978f8f;
```

```
    padding: 10px;
    border-radius: 50%;
    font-size: 24px;
    text-decoration: none;
    color: #333;
    box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.1);
  }
  #goHomeBtn:hover {
    background-color: #e0e0e0;
    cursor: pointer;
    transform: scale(1.05);
  }
  #toolsContainer {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 10px;
  }
  #shapesContainer {
    display: grid;
    grid-template-columns: repeat(7, 1fr);
    gap: 10px;
  }
  /* Toolbar */
  #toolbar {
    width: 70px;
    background-color: #dbbcbc;
    color: #fff;
    display: flex;
    flex-direction: column;
    justify-content: center;
    padding: 5px;
    gap: 1px;
    position: fixed;
    top: 0;
    bottom: 0;
    z-index: 10;
```

```
}
/* Tool Container */
.tool-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  position: relative;
}
.tool {
  width: 40px;
  height: 40px;
  display: flex;
  align-items: center;
  justify-content: center;
  background-color: #341d1d;
  border: none;
  border-radius: 30%;
  cursor: pointer;
  color: white;
  font-size: 20px;
  margin: 5px;
  position: relative;
  transition: cubic-bezier(0.075, 0.82, 0.165, 1);
}
.tool:hover {
  background-color: #777;
  transform: scale(1.05);
}
.tool.active {
  background-color: #6e6b6b;
}
/* Tool name shown on hover */
.tool:hover::after {
  content: attr(data-name);
  position: absolute;
  bottom: 100%;
```

```
    left: 50%;
    transform: translateX(-50%);
    background-color: rgba(0, 0, 0, 0.8);
    color: #fff;
    font-size: 12px;
    padding: 5px 10px;
    border-radius: 5px;
    white-space: nowrap;
    max-width: 120px;
    text-align: center;
    z-index: 20;
  }
  /* Zoom Buttons */
  .zoom-btn {
    background-color: #341d1d;
    border-radius: 5px;
    color: white;
    padding: 8px 12px;
    cursor: pointer;
    font-size: 14px;
  }

  .zoom-btn:hover {
    background-color: #777;
    transform: scale(1.05);
  }
  /* Exit Button */
  .exit-btn {
    background-color: red;
    color: white;
    font-size: 18px;
    border-radius: 50%;
    padding: 10px;
    cursor: pointer;
  }
  .exit-btn:hover {
```

```
    background-color: darkred;
    transform: scale(1.05);
  }
  /* Brush Options */
  #brushOptions {
    display: none;
    position: absolute;
    top: 100%;
    left: 50%;
    transform: translateX(-50%);
    background-color: #fff;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    border-radius: 8px;
    padding: 10px;
    z-index: 100;
  }
  #brushesContainer {
    position: relative;
  }
  #brushesContainer:hover #brushOptions {
    display: block;
    transform: scale(1.05);
  }
  #brushOptions button {
    margin: 5px;
  }
  #brushOptions button:hover::after {
    content: attr(title);
    position: absolute;
    bottom: 30px;
    left: 50%;
    transform: translateX(-50%);
    background-color: #000;
    color: #fff;
    padding: 2px 5px;
    border-radius: 3px;
```

```
    font-size: 14px;
    white-space: nowrap;
  }
  /* Sticky Notes */
  .sticky-note {
    position: absolute;
    background-color: #ffeb3b;
    padding: 10px;
    border-radius: 8px;
    width: 150px;
    height: 150px;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);
    cursor: move;
  }
  .sticky-note .note-text {
    min-height: 100%;
    white-space: pre-wrap;
    word-wrap: break-word;
    font-family: Arial, sans-serif;
  }
  .sticky-note .delete-btn {
    position: absolute;
    top: 5px;
    right: 5px;
    background: none;
    border: none;
    font-size: 20px;
    cursor: pointer;
  }
  .sticky-note .resize-handle {
    position: absolute;
    bottom: 5px;
    right: 5px;
    width: 15px;
    height: 15px;
    background: #000;
```

```
    cursor: se-resize;
  }
  .sticky-note.locked {
    pointer-events: none;
  }
  .sticky-note.dragging {
    opacity: 0.5;
  }
  #thumbnailPreview {
    max-width: 250px;
    max-height: 250px;

  }
```

```css
@media (max-width: 1000px) {
  #topRow {
    flex-direction: column;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 10px;
  }
  #toolsContainer, #shapesContainer {
    flex-direction: column;
    display: grid;
    justify-content: center;
    align-items: center;
    grid-template-columns: repeat(3, 1fr);
    gap: 5px;
  }
  #toolbar {
    width: 60px;
  }
  .iconBtn {
    width: 35px;
    height: 35px;
  }
  .tool {
    width: 35px;
    height: 35px;
  }
  #whiteboard {
    height: 400px;
  }
  #brushOptions {
    grid-template-columns: repeat(1, 1fr);
    padding: 4px;
    font-size: 10px;
    position: relative;
    top: 0;
    left: 0;
```

```css
    width: 100%;
  }
}

@media (max-width: 480px) {
  /* Containers */
  #toolsContainer, #shapesContainer {
    flex-direction: column;
    display: grid;
    justify-content: center;
    align-items: center;
    grid-template-columns: repeat(2, 1fr);
    gap: 5px;

  }
  #content {
    flex-direction: column;
    margin-left: 0;
    display: flex;
    align-items: center;
    justify-content: center;
  }
  #whiteboard-wrapper {
    width: 100%;
    height: 70%;
  }
  #whiteboard {
    height: 300px;
  }
  /* Icon Button */
  .iconBtn {
    background-color: #4f4040;
    border: none;
    color: white;
    font-size: 20px;
```

```css
    height: 40px;
    display: flex;
    align-items: center;
    justify-content: center;
    border-radius: 50%;
    cursor: pointer;
    position: relative;
    transition: transform 0.3s ease, background-color 0.3s ease;
  }
  /* Tool Button */
  .tool {
    width: 40px;
    height: 40px;
    display: flex;
    align-items: center;
    justify-content: center;
    background-color: #341d1d;
    border: none;
    border-radius: 30%;
    cursor: pointer;
    color: white;
    font-size: 20px;
    margin: 5px;
    position: relative;
    transition: transform 0.3s ease, cubic-bezier(0.075, 0.82, 0.165, 1);
  }
  #brushOptions {
    position: relative;
    top: 0;
    left: 0;
    width: 100%;
    transform: none;
    padding: 5px;
    font-size: 14px;
  }
```

```css
/* Animation effect */
@keyframes hoverEffect {
  0% {
    background-color: #f6e2e2;
  }
  100% {
    background-color: #aca9a9;
  }
}
.container:hover {
  animation: hoverEffect 0.3s forwards;
}
@keyframes iconHoverEffect {
```

```
  0% {
    transform: scale(1);
    background-color: #4f4040;
  }
  50% {
    transform: scale(1.2);
    background-color: #3d3232;
  }
  100% {
    transform: scale(1);
    background-color: #4f4040;
  }
}
.iconBtn:hover, .tool:hover, #goHomeBtn:hover {
  animation: iconHoverEffect 0.4s ease-in-out;
}
@keyframes fadeIn {
  0% {
    opacity: 0;
    transform: translateY(-10px);
  }
  100% {
    opacity: 1;
    transform: translateY(0);
  }
}
#toolbar , #topRow .iconBtn{
  animation: fadeIn 0.5s ease-out;
}
@keyframes whiteboardAnimation {
  0% {
    background-color: #f9f9f9;
    border-color: #e0e0e0;
    transform: scale(0.9);
    opacity: 0.8;
  }
```

```
  50% {
    background-color: #fff;
    border-color: #ccc;
    transform: scale(1.05);
    opacity: 1;
  }
  100% {
    background-color: #fff;
    border-color: #ccc;
    transform: scale(1);
    opacity: 1;
  }
}
#whiteboard {
  animation: whiteboardAnimation 1.5s ease-in-out;
}
```

Boardscript.js:

Java script code for New, Import, Save, Print, Exit, Zoom in, Zoom out, Clear, Grid lines, Full screen, Undo, Redo functionalities:

```
// Ensure DOM is fully loaded before running the script
document.addEventListener("DOMContentLoaded", () => {
  // Get elements for whiteboard tools
  const newBtn = document.getElementById("newBtn");
  const importBtn = document.getElementById("importBtn");
  const saveBtn = document.getElementById("saveBtn");
  const printBtn = document.getElementById("printBtn");
  const shareBtn = document.getElementById("shareBtn");
  const exitBtn = document.getElementById("exitBtn");
  const zoomInBtn = document.getElementById("zoomInBtn");
  const zoomOutBtn = document.getElementById("zoomOutBtn");
  const gridBtn = document.getElementById("gridBtn");
  const fullscreenBtn = document.getElementById("fullscreenBtn");
  const undoBtn = document.getElementById("undoBtn");
  const redoBtn = document.getElementById("redoBtn");

  // Get canvas elements
  const whiteboard = document.getElementById("whiteboard");
  const ctx = whiteboard.getContext("2d");
```

```
// New canvas
newBtn.addEventListener("click", () => {
  ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
  lines = [];
  undoneLines = [];
  if (showGrid) drawGrid();
});

// Import image
importBtn.addEventListener("click", () => {
  const input = document.createElement("input");
  input.type = "file";
  input.accept = "image/*";
  input.addEventListener("change", (e) => {
    const file = e.target.files[0];
    const reader = new FileReader();
    reader.onload = (e) => {
      const img = new Image();
      img.onload = () => {
        ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
        ctx.drawImage(img, 0, 0, whiteboard.width, whiteboard.height);
        if (showGrid) drawGrid();
      };
      img.src = e.target.result;
    };
    reader.readAsDataURL(file);
  });
  input.click();
});

// Save functionality
saveBtn.addEventListener("click", async () => {
  if (window.showSaveFilePicker) {
    try {
      const fileHandle = await window.showSaveFilePicker({
```

```
        suggestedName: "whiteboard.png",
        types: [
          { description: "PNG Image", accept: { "image/png": [".png"] } }
        ]
      });
      const writable = await fileHandle.createWritable();
      const dataUrl = whiteboard.toDataURL("image/png");
      const blob = await (await fetch(dataUrl)).blob();
      await writable.write(blob);
      await writable.close();
      alert("File saved successfully!");
    } catch (err) {
      console.error("Save operation failed:", err);
    }
  } else {
    alert("The File System Access API is not supported in this browser.");
  }
});

// Print canvas content
printBtn.addEventListener("click", () => {
  const dataUrl = whiteboard.toDataURL("image/png");
  const printWindow = window.open();
  printWindow.document.write("<img src='" + dataUrl + "' />");
  printWindow.print();
});

// Share canvas content
shareBtn.addEventListener("click", () => {
  const dataUrl = whiteboard.toDataURL("image/png");
  navigator.share({ files: [new File([dataUrl], "whiteboard.png", { type: "image/png" })] });
});
```

```
// Exit functionality
exitBtn.addEventListener("click", () => window.close());

// Fullscreen toggle
fullscreenBtn.addEventListener("click", () => {
  if (document.fullscreenElement) {
    document.exitFullscreen();
  } else {
    document.documentElement.requestFullscreen();
  }
});

// Zoom functionality
zoomInBtn.addEventListener("click", () => {
  zoomLevel += 0.1;
  whiteboard.style.transform = `scale(${zoomLevel})`;
});

zoomOutBtn.addEventListener("click", () => {
  zoomLevel = Math.max(0.1, zoomLevel - 0.1);
  whiteboard.style.transform = `scale(${zoomLevel})`;
});

  let currentLine = null;
  // Draw gridlines on the canvas
  function drawGrid() {
    const gridSize = 20;
    ctx.strokeStyle = "#ddd";
    for (let x = 0; x < whiteboard.width; x += gridSize) {
      ctx.beginPath();
      ctx.moveTo(x, 0);
      ctx.lineTo(x, whiteboard.height);
      ctx.stroke();
    }
    for (let y = 0; y < whiteboard.height; y += gridSize) {
      ctx.beginPath();
```

```
      ctx.moveTo(0, y);
      ctx.lineTo(whiteboard.width, y);
      ctx.stroke();
    }
  }

  // Redraw the canvas
  function redrawCanvas() {
    ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
    if (showGrid) drawGrid();
    lines.forEach((line) => {
      ctx.beginPath();
      ctx.moveTo(line.x1, line.y1);
      ctx.lineTo(line.x2, line.y2);
      ctx.strokeStyle = line.color;
      ctx.lineWidth = line.width;
      ctx.stroke();
    });
  }

  // Start drawing
  whiteboard.addEventListener("mousedown", (e) => {
    drawing = true;
    currentLine = {
      startX: e.offsetX,
      startY: e.offsetY,
      color: "#fff",
      width:4 ,
    };
    undoneLines = []; // Clear redo history when starting a new drawing
  });

  // Stop drawing
  whiteboard.addEventListener("mouseup", () => {
    drawing = false;
```

```
    drawing = false;
    currentLine = null;
  });

  // Undo functionality
  undoBtn.addEventListener("click", () => {
    if (lines.length > 0) {
      const lastLine = lines.pop();
      undoneLines.push(lastLine); // Save the undone line for redo
      redrawCanvas();
    }
  });

  // Redo functionality
  redoBtn.addEventListener("click", () => {
    if (undoneLines.length > 0) {
      const lastUndoneLine = undoneLines.pop();
      lines.push(lastUndoneLine); // Restore the undone line
      redrawCanvas();
    }
  });

  // Toggle gridlines
  gridBtn.addEventListener("click", () => {
    showGrid = !showGrid;
    redrawCanvas();
  });

  // Resize the canvas on window resize
  window.addEventListener("resize", resizeCanvas);
  resizeCanvas();
```

```
//clear functionality
const clearBtn = document.getElementById('clearBtn');
let isSelecting = false;
let selectedObject = null;

// Get references to the canvas and the thumbnail image
const thumbnailPreview = document.getElementById('thumbnailPreview');
// Function to capture the thumbnail
function captureThumbnail() {

  if (whiteboard) {
    const thumbnail = whiteboard.toDataURL('image/png');
    // Set the thumbnail as the source for the preview image
    thumbnailPreview.src = thumbnail;
    thumbnailPreview.style.display = 'block'; // Make sure the thumbnail is visible
  }
}

whiteboard.addEventListener('mouseup', captureThumbnail);
whiteboard.addEventListener('touchend', captureThumbnail);

// Function to clear the whiteboard
clearBtn.addEventListener('click', () => {
  const ctx = whiteboard.getContext('2d');
  ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
});
```

Java script code for Pencil, Eraser, text and Marker tools functionalities:

```
// Initialize canvas context
let zoomLevel = 1;
let lastX = 0;
let lastY = 0;
let lines = [];
let undoneLines = [];
let showGrid = false;

let currentTool = 'pencil'; // Track the selected tool
let drawing = false;
let isErasing = false;

// Tool Buttons
const pencilBtn = document.getElementById("pencilBtn");
const eraserBtn = document.getElementById("eraserBtn");
const textBtn = document.getElementById("textBtn");
const markerBtn = document.getElementById("markerBtn");

// Tool Selection Handlers
pencilBtn.addEventListener("click", () => {
  currentTool = 'pencil';
  isErasing = false;
  ctx.strokeStyle = "#000";
  ctx.lineWidth = 2;
});

let eraserSize = 50;

// Event listener for the eraser button
eraserBtn.addEventListener("click", () => {
  currentTool = 'eraser';
  ctx.globalCompositeOperation = 'destination-out'; // Set to eraser mode
});
```

```
// Erase function to clear an area
function erase(x, y) {
  ctx.beginPath();
  ctx.arc(x, y, eraserSize / 2, 0, Math.PI * 2); // Create a circular eraser
  ctx.fill(); // Clear the area inside the circle
}

// Mouse events for erasing
whiteboard.addEventListener("mousemove", (e) => {
  if (currentTool === 'eraser' && drawing) {
    erase(e.offsetX, e.offsetY);
  }
});

whiteboard.addEventListener("mouseup", () => {
  if (currentTool === 'eraser') {
    ctx.globalCompositeOperation = 'source-over'; // Reset to normal mode
  }
});

markerBtn.addEventListener("click", () => {
  currentTool = 'marker';
  isErasing = false;
  ctx.strokeStyle = "#FF0000";
  ctx.lineWidth = 5;
});

// Event listener for the text tool button
textBtn.addEventListener("click", () => {
  currentTool = 'text';
});

// Mouse Events for Drawing/Erasing/Text
whiteboard.addEventListener("mousedown", (e) => {
```

```
whiteboard.addEventListener("mousedown", (e) => {
  drawing = true;
  lastX = e.offsetX;
  lastY = e.offsetY;

  if (currentTool === 'text') {
    drawing = false; // Stop drawing when using the text tool
    const text = prompt("Enter text:");
    if (text) {
      ctx.font = "16px Arial"; // Font size and style
      ctx.fillStyle = ctx.strokeStyle; // Use current stroke style for text color
      ctx.fillText(text, e.offsetX, e.offsetY); // Draw text at the clicked position
    }
  }
});

whiteboard.addEventListener("mousemove", (e) => {
  if (!drawing || currentTool === 'text') return;

  const currentX = e.offsetX;
  const currentY = e.offsetY;

  if (isErasing) {
    erase(currentX, currentY); // Use existing erase function
  } else {
    ctx.beginPath();
    ctx.moveTo(lastX, lastY);
    ctx.lineTo(currentX, currentY);
    ctx.stroke();

    // Save the line for undo/redo functionality
    lines.push({ x1: lastX, y1: lastY, x2: currentX, y2: currentY });
  }

  lastX = currentX;
```

```
    lastY = currentY;
  });

  // Stop drawing when mouse is released
  whiteboard.addEventListener("mouseup", () => {
    drawing = false;
  });

  // Ensure the drawing stops when the mouse leaves the canvas
  whiteboard.addEventListener("mouseout", () => {
    drawing = false;
  });

  // Resize canvas
  function resizeCanvas() {
    whiteboard.width = window.innerWidth - 80;
    whiteboard.height = window.innerHeight * 0.8;
    if (showGrid) drawGrid(); // Redraw the grid after resizing
  }
  resizeCanvas();

  // Clear canvas
  function clearCanvas() {
    ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
  }

```

Java script code for shapes functionality:

```
// Store drawn shapes in an array
let shapes = [];
let currentShape = null;
let drawingShape = false;
let startX, startY;

// Event listeners for shape buttons
document.getElementById("squareBtn").addEventListener("click", () => setCurrentShape("square"));
document.getElementById("circleBtn").addEventListener("click", () => setCurrentShape("circle"));
document.getElementById("triangleBtn").addEventListener("click", () => setCurrentShape("triangle"));
document.getElementById("ellipseBtn").addEventListener("click", () => setCurrentShape("ellipse"));
document.getElementById("hexagonBtn").addEventListener("click", () => setCurrentShape("hexagon"));
document.getElementById("starBtn").addEventListener("click", () => setCurrentShape("star"));
document.getElementById("pentagonBtn").addEventListener("click", () => setCurrentShape("pentagon"));
document.getElementById("rhombusBtn").addEventListener("click", () => setCurrentShape("rhombus"));
document.getElementById("parallelogramBtn").addEventListener("click", () => setCurrentShape("parallelogram"));
document.getElementById("octagonBtn").addEventListener("click", () => setCurrentShape("octagon"));
document.getElementById("crossBtn").addEventListener("click", () => setCurrentShape("cross"));
document.getElementById("heartBtn").addEventListener("click", () => setCurrentShape("heart"));
document.getElementById("lineBtn").addEventListener("click", () => setCurrentShape("line"));
document.getElementById("arrowBtn").addEventListener("click", () => setCurrentShape("arrow"));

// Function to set the current shape to draw
function setCurrentShape(shape) {
  currentShape = shape;
  drawingShape = false;
}

// Event listeners for mouse actions
whiteboard.addEventListener("mousedown", (e) => {
  if (!currentShape) return;
  drawingShape = true;
  startX = e.offsetX;
  startY = e.offsetY;
});
```

```
whiteboard.addEventListener("mousemove", (e) => {
  if (!drawingShape) return;
  const endX = e.offsetX;
  const endY = e.offsetY;
  clearCanvas();
  redrawShapes();
  drawShape(currentShape, startX, startY, endX, endY);
});

whiteboard.addEventListener("mouseup", (e) => {
  if (!drawingShape) return;
  drawingShape = false;
  const endX = e.offsetX;
  const endY = e.offsetY;
  shapes.push({ shape: currentShape, startX, startY, endX, endY }); // Store the shape data
  redrawShapes();
});

// Function to draw different shapes
function drawShape(shape, x1, y1, x2, y2) {
  const width = x2 - x1;
  const height = y2 - y1;
  ctx.beginPath();
  switch (shape) {
    case "square":
      ctx.rect(x1, y1, width, width);
      break;
    case "circle":
      const radius = Math.sqrt(width * width + height * height) / 2;
      ctx.arc(x1 + width / 2, y1 + height / 2, radius, 0, Math.PI * 2);
      break;
    case "triangle":
      ctx.moveTo(x1, y2);
      ctx.lineTo(x1 + width / 2, y1);
      ctx.lineTo(x2, y2);
```

```
      ctx.closePath();
      break;
    case "ellipse":
      ctx.ellipse(x1 + width / 2, y1 + height / 2, Math.abs(width) / 2, Math.abs(height) / 2, 0, 0, Math.PI * 2);
      break;
    case "hexagon":
      drawPolygon(6, x1, y1, width, height);
      break;
    case "pentagon":
      drawPolygon(5, x1, y1, width, height);
      break;
    case "star":
      drawStar(x1 + width / 2, y1 + height / 2, Math.abs(width) / 2, Math.abs(width) / 4, 5);
      break;
    case "rhombus":
      ctx.moveTo(x1 + width / 2, y1);
      ctx.lineTo(x2, y1 + height / 2);
      ctx.lineTo(x1 + width / 2, y2);
      ctx.lineTo(x1, y1 + height / 2);
      ctx.closePath();
      break;
    case "parallelogram":
      ctx.moveTo(x1 + width / 4, y1);
      ctx.lineTo(x2, y1);
      ctx.lineTo(x2 - width / 4, y2);
      ctx.lineTo(x1, y2);
      ctx.closePath();
      break;
    case "octagon":
      drawPolygon(8, x1, y1, width, height);
      break;
    case "cross":
      drawCross(x1, y1, width, height);
      break;
```

```
      case "heart":
        drawHeart(x1, y1, width, height);
        break;
      case "line":
        ctx.moveTo(x1, y1);
        ctx.lineTo(x2, y2);
        break;
      case "arrow":
        drawArrow(x1, y1, x2, y2);
        break;
    }
    ctx.stroke();
  }

  // Function to draw a polygon (for hexagons, pentagons, etc.)
  function drawPolygon(sides, x1, y1, width, height) {
    const centerX = x1 + width / 2;
    const centerY = y1 + height / 2;
    const radius = Math.min(Math.abs(width), Math.abs(height)) / 2;
    ctx.moveTo(centerX + radius * Math.cos(0), centerY + radius * Math.sin(0));
    for (let i = 1; i <= sides; i++) {
      const angle = (i * 2 * Math.PI) / sides;
      ctx.lineTo(centerX + radius * Math.cos(angle), centerY + radius * Math.sin(angle));
    }
    ctx.closePath();
  }

  // Function to draw a star
  function drawStar(cx, cy, outerRadius, innerRadius, points) {
    const step = Math.PI / points;
    let angle = -Math.PI / 2;
    ctx.moveTo(cx + outerRadius * Math.cos(angle), cy + outerRadius * Math.sin(angle));
    for (let i = 0; i < points; i++) {
      angle += step;
      ctx.lineTo(cx + innerRadius * Math.cos(angle), cy + innerRadius * Math.sin(angle));
      angle += step;
      ctx.lineTo(cx + outerRadius * Math.cos(angle), cy + outerRadius * Math.sin(angle));
    }
    ctx.closePath();
  }

  // Function to draw a cross
  function drawCross(x1, y1, width, height) {
    const armWidth = Math.min(width, height) / 4;
    const centerX = x1 + width / 2;
    const centerY = y1 + height / 2;
    // Horizontal line
    ctx.moveTo(centerX - armWidth / 2, centerY);
    ctx.lineTo(centerX + armWidth / 2, centerY);
    // Vertical line
    ctx.moveTo(centerX, centerY - armWidth / 2);
    ctx.lineTo(centerX, centerY + armWidth / 2);
  }

  // Function to draw a heart
  function drawHeart(x1, y1, width, height) {
    const x2 = x1 + width;
    const y2 = y1 + height;
    ctx.moveTo(x1 + width / 2, y1 + height / 4);
    ctx.bezierCurveTo(x1 + width / 2, y1, x1, y1, x1, y1 + height / 4);
    ctx.bezierCurveTo(x1, y1 + height / 2, x1 + width / 2, y2, x1 + width / 2, y2);
    ctx.bezierCurveTo(x1 + width / 2, y2, x2, y1 + height / 2, x2, y1 + height / 4);
    ctx.bezierCurveTo(x2, y1, x1 + width / 2, y1, x1 + width / 2, y1 + height / 4);
    ctx.closePath();
  }

  // Function to draw an arrow
  function drawArrow(x1, y1, x2, y2) {
    ctx.moveTo(x1, y1);
    ctx.lineTo(x2, y2);
```

```
    // Arrowhead
    const angle = Math.atan2(y2 - y1, x2 - x1);
    const arrowLength = 15;  // Length of the arrowhead
    const arrowAngle = Math.PI / 6;  // Angle of the arrowhead
    ctx.lineTo(x2 - arrowLength * Math.cos(angle - arrowAngle), y2 - arrowLength * Math.sin(angle - arrowAngle));
    ctx.moveTo(x2, y2);
    ctx.lineTo(x2 - arrowLength * Math.cos(angle + arrowAngle), y2 - arrowLength * Math.sin(angle + arrowAngle));
    ctx.stroke();
  }

  // Function to redraw all the shapes
  function redrawShapes() {
    shapes.forEach((shapeData) => {
      drawShape(shapeData.shape, shapeData.startX, shapeData.startY, shapeData.endX, shapeData.endY);
    });
  }

  // Function to clear the canvas
  function clearCanvas() {
    ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
    if (showGrid) drawGrid(); // Redraw grid if enabled
  }
```

Java script code for Color Picker functionality:

```
    // Function to change the brush color based on the color picker
   function changeBrushColor(color) {
     ctx.strokeStyle = color;
     ctx.fillStyle = color;
   }
     // Event listener for color picker
     colorPicker.addEventListener("input", () => {
       setBrushSettings(currentBrush); // Update brush settings when color is changed
     });

     // Mouse events for drawing with brush
     whiteboard.addEventListener("mousedown", (e) => {
       drawing = true;
       lastX = e.offsetX;
       lastY = e.offsetY;
       ctx.beginPath();
       ctx.moveTo(lastX, lastY);
     });

     whiteboard.addEventListener("mouseup", () => {
       drawing = false;
       ctx.closePath();
     });

     whiteboard.addEventListener("mouseout", () => {
       drawing = false;
       ctx.closePath();
     });

   pencilBtn.addEventListener("click", () => {
     currentTool = 'pencil';
     isErasing = false;
     ctx.strokeStyle = document.getElementById("color-picker").value; // Set to selected color
     ctx.lineWidth = 2; // Thin line for pencil
   });
```

```
  markerBtn.addEventListener("click", () => {
    currentTool = 'marker';
    isErasing = false;
    ctx.strokeStyle = document.getElementById("color-picker").value; // Set to selected color
    ctx.lineWidth = 5; // Thick line for marker
  });

```

Java Script code for Brushes functionalities:

```
// Brushes options
const crayonBtn = document.getElementById("crayonBtn");
const airbrushBtn = document.getElementById("airbrushBtn");
const watercolorBtn = document.getElementById("watercolorBtn");
const oilBrushBtn = document.getElementById("oilBrushBtn");
const calligraphyBtn = document.getElementById("calligraphyBtn");
const colorPicker = document.getElementById("color-picker");

// Set up brush tool properties
const brushSettings = {
  crayon: { lineWidth: 15, lineCap: "round", globalAlpha: 1 },
  airbrush: { lineWidth: 5, lineCap: "round", globalAlpha: 0.3 },
  watercolor: { lineWidth: 10, lineCap: "round", globalAlpha: 0.5 },
  oilBrush: { lineWidth: 20, lineCap: "round", globalAlpha: 1 },
  calligraphy: { lineWidth: 5, lineCap: "butt", globalAlpha: 1 }
};

// Set brush settings based on selected tool
function setBrushSettings(brushType) {
  const settings = brushSettings[brushType];
  ctx.lineWidth = settings.lineWidth;
  ctx.lineCap = settings.lineCap;
  ctx.globalAlpha = settings.globalAlpha;
  ctx.strokeStyle = colorPicker.value;
}

// Set default brush settings
let currentBrush = "crayon";
setBrushSettings(currentBrush);

// Brush tool event listeners
crayonBtn.addEventListener("click", () => {
  currentBrush = "crayon";
  setBrushSettings(currentBrush);
});
```

```
airbrushBtn.addEventListener("click", () => {
  currentBrush = "airbrush";
  setBrushSettings(currentBrush);
});

watercolorBtn.addEventListener("click", () => {
  currentBrush = "watercolor";
  setBrushSettings(currentBrush);
});

oilBrushBtn.addEventListener("click", () => {
  currentBrush = "oilBrush";
  setBrushSettings(currentBrush);
});

calligraphyBtn.addEventListener("click", () => {
  currentBrush = "calligraphy";
  setBrushSettings(currentBrush);
});
```

Java Script code for Sticky notes functionality:

```
//sticky notes
let stickyNotes = [];
let selectedStickyNote = null;
let offsetX = 0;
let offsetY = 0;

document.getElementById('addStickyNoteBtn').addEventListener('click', function () {
  addStickyNote();
});

// Add a sticky note to the canvas
function addStickyNote() {
  const note = {
    id: Date.now(),
    x: 100, // Initial position
    y: 100,
    width: 150,
    height: 150,
    text: 'New Note',
    isEditing: false
  };
  stickyNotes.push(note);
  renderStickyNotes();
}

// Render all sticky notes on the canvas
function renderStickyNotes() {
  const canvas = document.getElementById('whiteboard');
  const ctx = canvas.getContext('2d');
  ctx.clearRect(0, 0, canvas.width, canvas.height); // Clear the canvas
  stickyNotes.forEach(note => {
    ctx.fillStyle = '#FFD700';
    ctx.fillRect(note.x, note.y, note.width, note.height);
    ctx.fillStyle = 'black';
    ctx.font = '16px Arial';
    wrapText(ctx, note.text, note.x + 10, note.y + 30, note.width - 20, note.height - 40);
  });
}

// Wrap text inside the sticky note box, ensuring it fits within the box boundaries
function wrapText(ctx, text, x, y, maxWidth, maxHeight) {
  const words = text.split(' ');
  let line = '';
  let lineHeight = 20;
  let currentHeight = y;
  for (let i = 0; i < words.length; i++) {
    const testLine = line + words[i] + ' ';
    const testWidth = ctx.measureText(testLine).width;
    // If the line exceeds the maxWidth, draw the current line and start a new one
    if (testWidth > maxWidth && i > 0) {
      ctx.fillText(line, x, currentHeight);
      line = words[i] + ' ';
      currentHeight += lineHeight;
    } else {
      line = testLine;
    }
    // If the current height exceeds the max height, stop drawing more text
    if (currentHeight + lineHeight > y + maxHeight) {
      ctx.fillText('...', x, currentHeight); // Indicate that the text is truncated
      break;
    }
  }
  // Draw the last line of text if within bounds
  if (currentHeight + lineHeight <= y + maxHeight) {
    ctx.fillText(line, x, currentHeight);
  }
}
```

```
// Edit a sticky note
function editStickyNote(note) {
  const newText = prompt('Edit Sticky Note:', note.text);
  if (newText !== null) {
    note.text = newText;
    renderStickyNotes();
  }
}
// Delete a sticky note
function deleteStickyNote(note) {
  const index = stickyNotes.indexOf(note);
  if (index > -1) {
    stickyNotes.splice(index, 1);
    renderStickyNotes();
  }
}

// Event listeners for interacting with sticky notes
document.getElementById('whiteboard').addEventListener('mousedown', function (e) {
  const canvas = e.target;
  const rect = canvas.getBoundingClientRect();
  const x = e.clientX - rect.left;
  const y = e.clientY - rect.top;

  // Check if a sticky note is clicked
  for (let i = stickyNotes.length - 1; i >= 0; i--) {
    const note = stickyNotes[i];
    if (x >= note.x && x <= note.x + note.width && y >= note.y && y <= note.y + note.height) {
      selectedStickyNote = note;
      offsetX = x - note.x;
      offsetY = y - note.y;

      // If Ctrl key is pressed, delete the sticky note
      if (e.ctrlKey) {
        deleteStickyNote(note);
      } else {
        // If the sticky note is clicked, edit it
        if (!note.isEditing) {
          note.isEditing = true;
          editStickyNote(note);
        }
      }
      return;
    }
  }
});
// Move sticky note when dragging
document.getElementById('whiteboard').addEventListener('mousemove', function (e) {
  if (selectedStickyNote) {
    const canvas = e.target;
    const rect = canvas.getBoundingClientRect();
    const x = e.clientX - rect.left;
    const y = e.clientY - rect.top;
    selectedStickyNote.x = x - offsetX;
    selectedStickyNote.y = y - offsetY;
    renderStickyNotes();
  }
});

// Release selected sticky note when mouse button is released
document.getElementById('whiteboard').addEventListener('mouseup', function () {
  selectedStickyNote = null;
});

// Initial rendering
renderStickyNotes();
```

JavaScript code for Background functionality:

```
//Backgroung Btn
const backgroundColorBtn = document.getElementById("backgroundColorBtn");

// Initialize the whiteboard canvas
const canvas = whiteboard;
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
// Function to change the background color
function changeBackgroundColor() {
    // Open a color picker dialog
    const colorPicker = document.createElement("input");
    colorPicker.type = "color";
    colorPicker.style.display = "none";
    document.body.appendChild(colorPicker);
    colorPicker.click();

    // Update the canvas background color when a color is selected
    colorPicker.addEventListener("input", (event) => {
        const selectedColor = event.target.value;
        ctx.fillStyle = selectedColor;
        ctx.fillRect(0, 0, canvas.width, canvas.height);
    });

    // Clean up the color picker element
    colorPicker.addEventListener("change", () => {
        document.body.removeChild(colorPicker);
    });
}
backgroundColorBtn.addEventListener("click", changeBackgroundColor);
```

JavaScript code for Export functionality:

```
//Export functionality
document.getElementById("exportPdfBtn").addEventListener("click", function () {
  const canvas = document.getElementById("whiteboard");
  if (!canvas) {
    alert("Whiteboard canvas not found!");
    return;
  }
  // Convert canvas to an image
  const canvasData = canvas.toDataURL("image/png");
  // Initialize jsPDF
  const { jsPDF } = window.jspdf;
  const pdf = new jsPDF();
  // Get the canvas dimensions
  const canvasWidth = canvas.width;
  const canvasHeight = canvas.height;
  // Calculate PDF dimensions (A4 size: 210mm x 297mm)
  const pageWidth = pdf.internal.pageSize.getWidth();
  const pageHeight = (canvasHeight / canvasWidth) * pageWidth;
  // Add the image to the PDF
  pdf.addImage(canvasData, "PNG", 0, 0, pageWidth, pageHeight);
  // Prompt the user to download the PDF
  pdf.save("whiteboard.pdf");
});
// Selecting necessary elements
const lineWidthSlider = document.getElementById('lineWidth');
// Canvas and Context setup
canvas.width = canvas.offsetWidth;
canvas.height = canvas.offsetHeight;
// Default settings
let currentLineWidth = 2;
// Update line width based on slider value
lineWidthSlider.addEventListener('input', (e) => {
  currentLineWidth = e.target.value;
  ctx.lineWidth = currentLineWidth;
});
```

```
// Tool selection logic
function selectTool(toolName) {
  currentTool = toolName;
  switch (toolName) {
    case 'eraser':
      ctx.globalCompositeOperation = 'destination-out';
      break;
    default:
      ctx.globalCompositeOperation = 'source-over';
      break;
  }
  ctx.lineWidth = currentLineWidth;
}

// Tool button event listeners
pencilBtn.addEventListener('click', () => selectTool('pencil'));
eraserBtn.addEventListener('click', () => selectTool('eraser'));
markerBtn.addEventListener('click', () => selectTool('marker'));
crayonBtn.addEventListener('click', () => selectTool('crayon'));
airbrushBtn.addEventListener('click', () => selectTool('airbrush'));
oilBrushBtn.addEventListener('click', () => selectTool('oilBrush'));
calligraphyBtn.addEventListener('click', () => selectTool('calligraphy'));
watercolorBtn.addEventListener('click', () => selectTool('watercolor'));

// Drawing functionality
let isDrawing = false;

let isFirstMove = true;
// Smooth drawing
function drawLine(x, y) {
  if (isFirstMove) {
    ctx.moveTo(x, y);
    isFirstMove = false;
  } else {
    ctx.lineTo(x, y);
    ctx.stroke();
```

```
    }
  }
  // Drawing event listeners
  canvas.addEventListener('mousedown', (e) => {
    isDrawing = true;
    const rect = canvas.getBoundingClientRect();
    lastX = e.clientX - rect.left;
    lastY = e.clientY - rect.top;
    ctx.beginPath();
    isFirstMove = true;   // Reset on mouse down
  });

  canvas.addEventListener('mousemove', (e) => {
    if (!isDrawing) return;

    const rect = canvas.getBoundingClientRect();
    const x = e.clientX - rect.left;
    const y = e.clientY - rect.top;

    drawLine(x, y);
  });

  canvas.addEventListener('mouseup', () => {
    isDrawing = false;
    lastX = 0;
    lastY = 0;
    isFirstMove = true;
  });

  canvas.addEventListener('mouseout', () => {
    isDrawing = false;
    lastX = 0;
    lastY = 0;
    isFirstMove = true;
  });
```

6. Challenges Faced

1. Implementing accurate and responsive shape drawing tools (such as Square, Circle, Triangle, etc.) posed difficulties, especially in terms of precision and adjusting shapes dynamically on the canvas.
2. Allowing users to add, move, delete and add text into sticky note on the whiteboard was complex, as it required precise positioning and interaction handling without interfering with the drawing tools.

7. Future Enhancements

1. Collaboration Features

   Enhancement: Allow multiple users to collaborate on the whiteboard in real-time, enabling features like live drawing, chat, and the ability to see other users' actions.

2. Cloud Storage Integration

   Enhancement: Enable users to save their projects directly to cloud storage (e.g., Google Drive, Dropbox) for easy access and backup. Enhanced Cart features and also made product review form.

3. Interactive Templates

   Enhancement: Provide a library of pre-designed templates (e.g., mind maps, flowcharts, presentation slides) that users can select and modify.

4. User Profiles and Personalization

   Enhancement: Allow users to create profiles and personalize their whiteboard interface with themes, preferred tools, and saved settings.

8. Conclusion

The Digital Whiteboard project successfully combines interactive drawing tools, a user-friendly interface, and a range of customizable features to provide a versatile platform for digital collaboration and creative expression. By leveraging HTML, CSS, and JavaScript, the whiteboard allows users to seamlessly draw, write, and manipulate content, offering a responsive and dynamic experience across devices.

9. References

1. HTML DOM: Retrieved from https://www.w3schools.com/js/js_htmldom.asp

2. CSS-Tricks: A Complete Guide to Grid. Retrieved from https://css-tricks.com/snippets/css/complete-guide-grid/

3. Event Handling: Retrieved from https://javascript.info/introduction-browser-events

4. A List Apart: Responsive Web Design. Retrieved from https://alistapart.com/article/responsive-web-design/