

**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
2. Project Overview
3. Technologies Used
4. System Architecture
5. Implementation Details
6. Challenges Faced
7. Future Enhancements
8. Conclusion
9. References

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 `` 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.

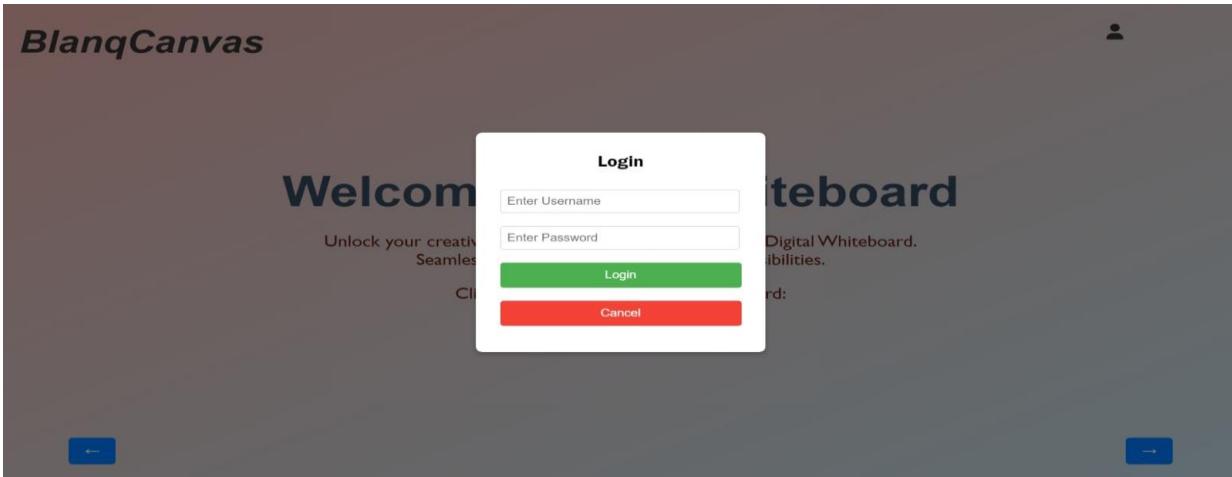
Fig 1: Home page



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.

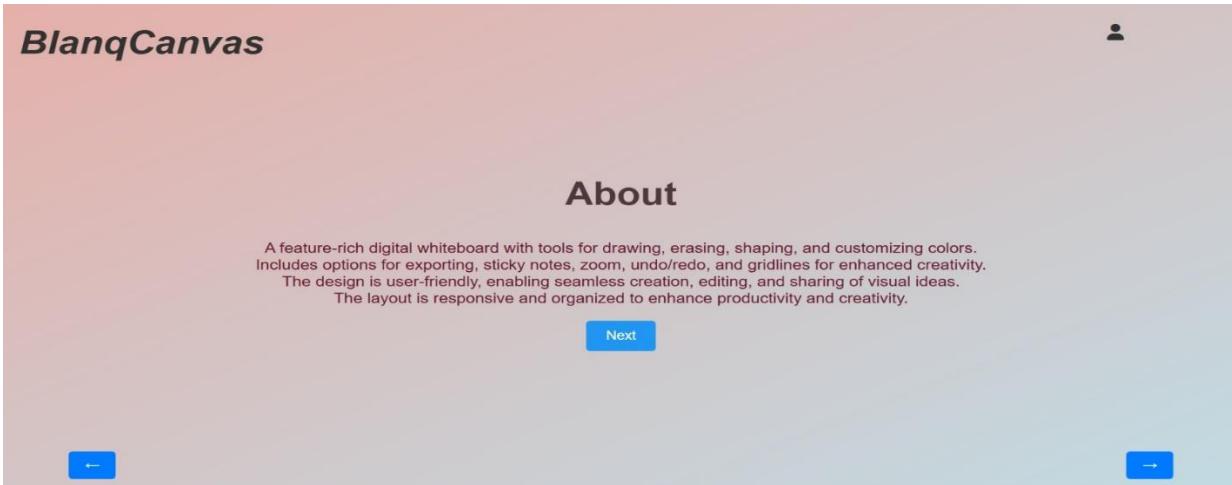
Fig 2: Login



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

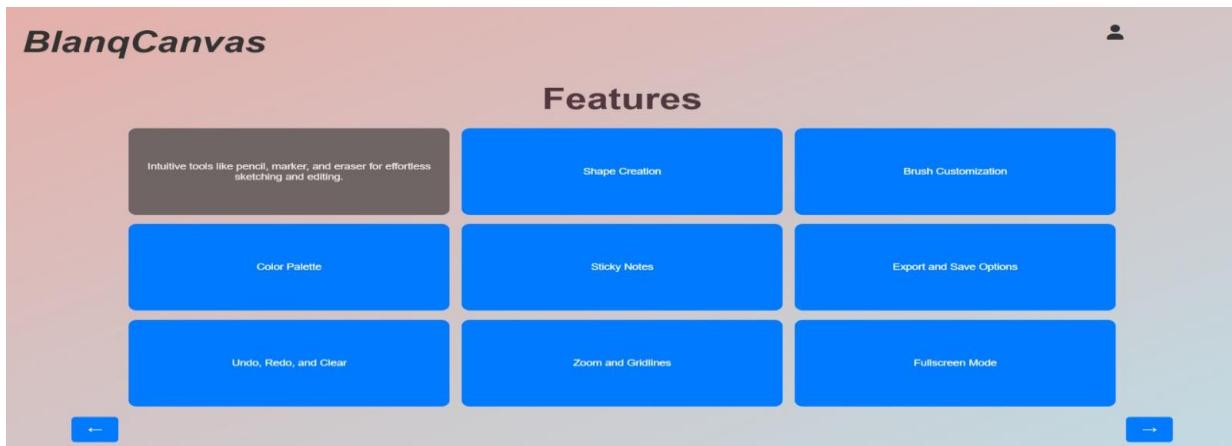
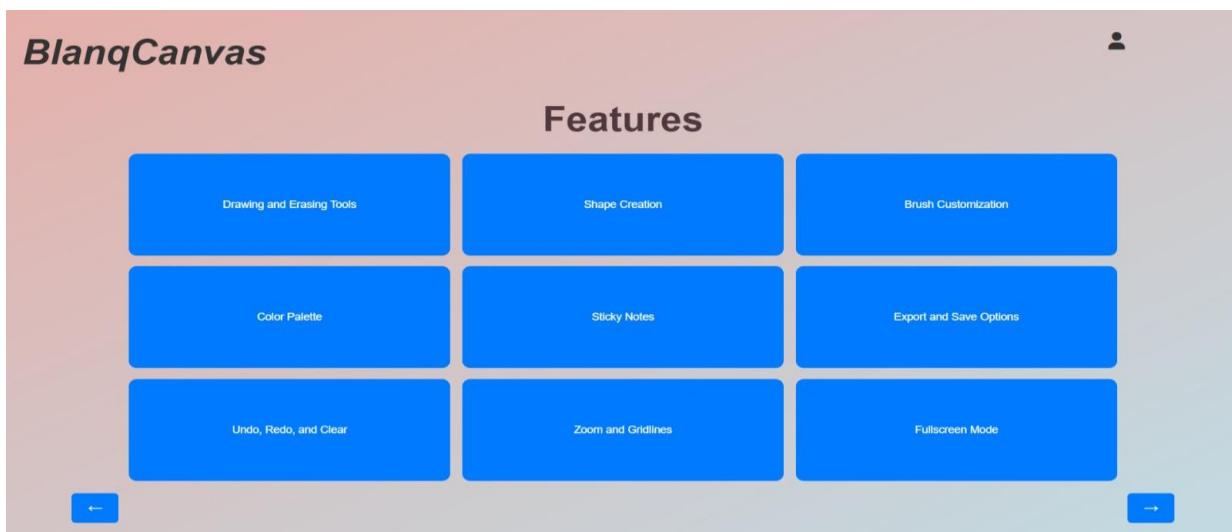


Features:

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.

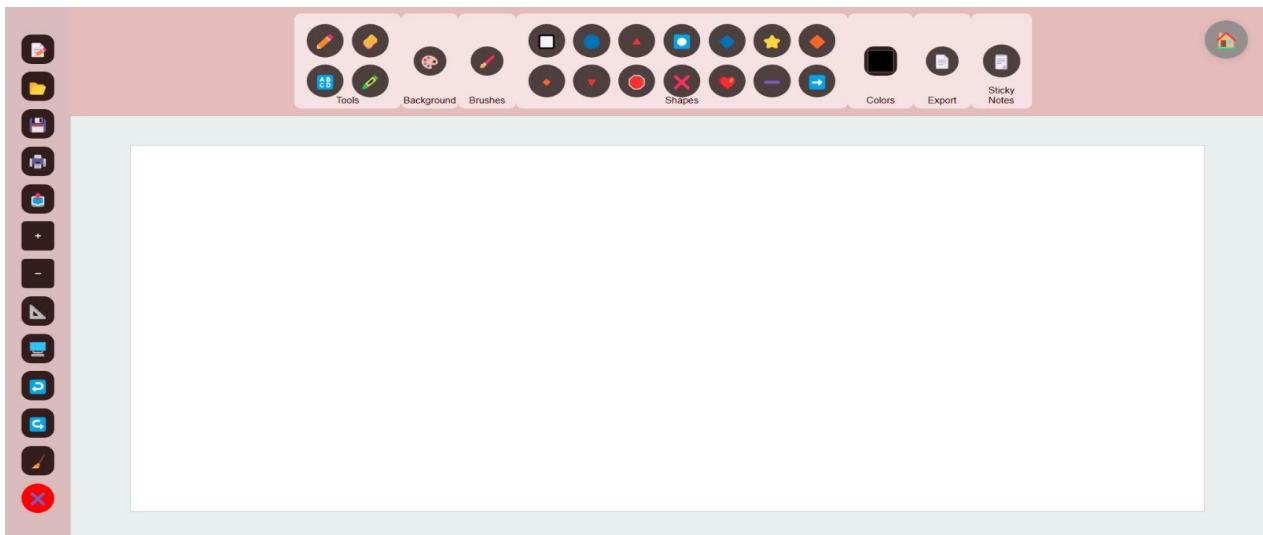
Technologies Used:

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

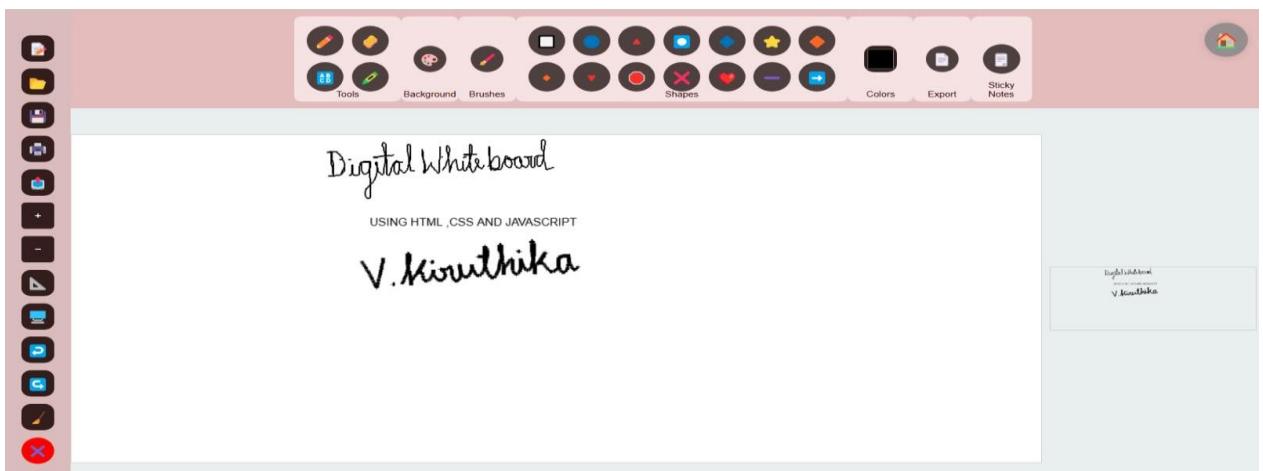
CSS: Styles the user interface and ensures the responsiveness of the whiteboard on different devices.

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.

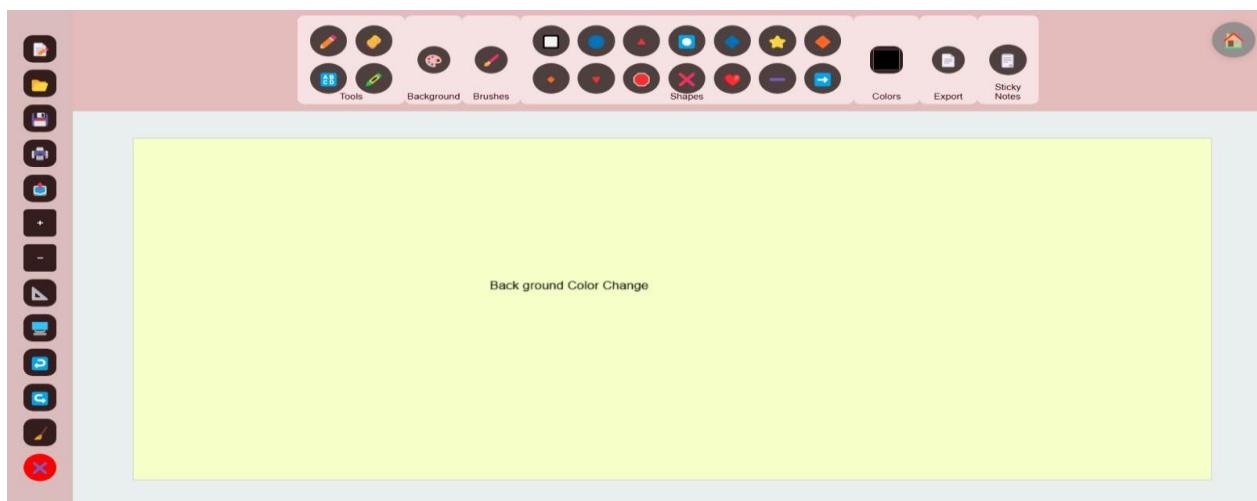
Fig 5: Digital White Board



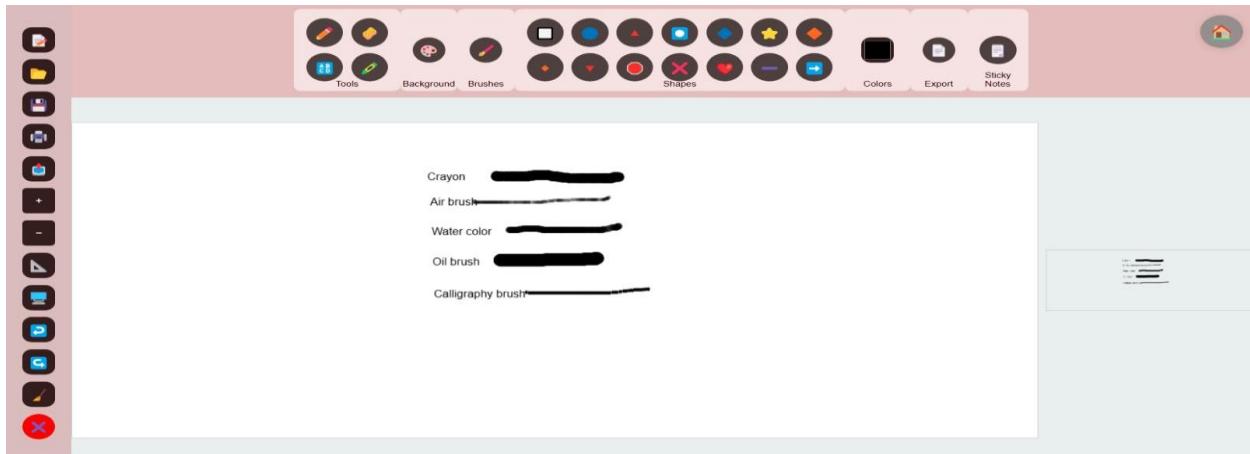
Tools:



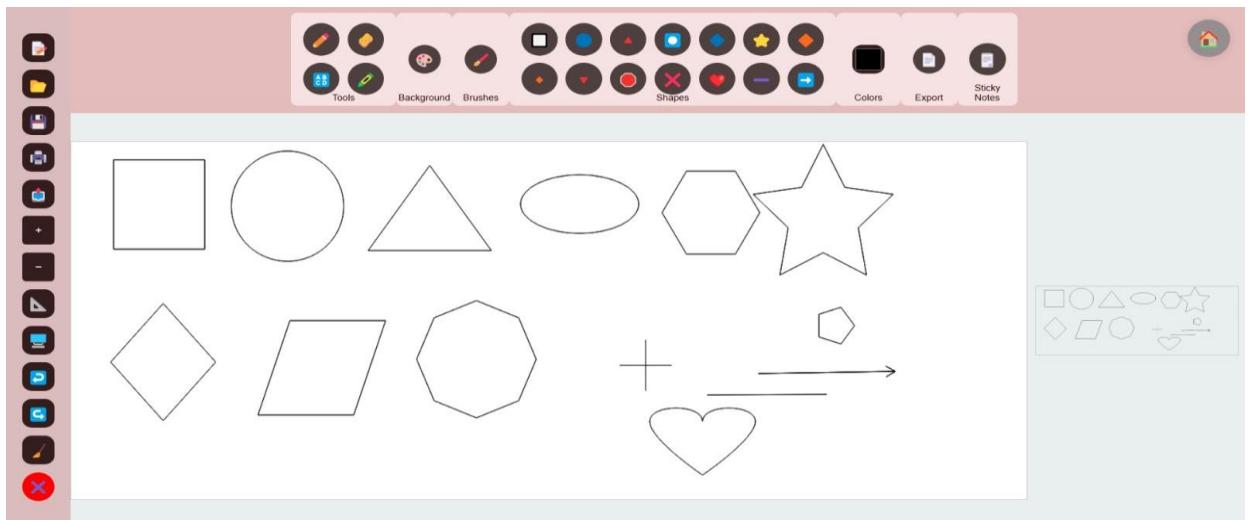
Back ground Color:



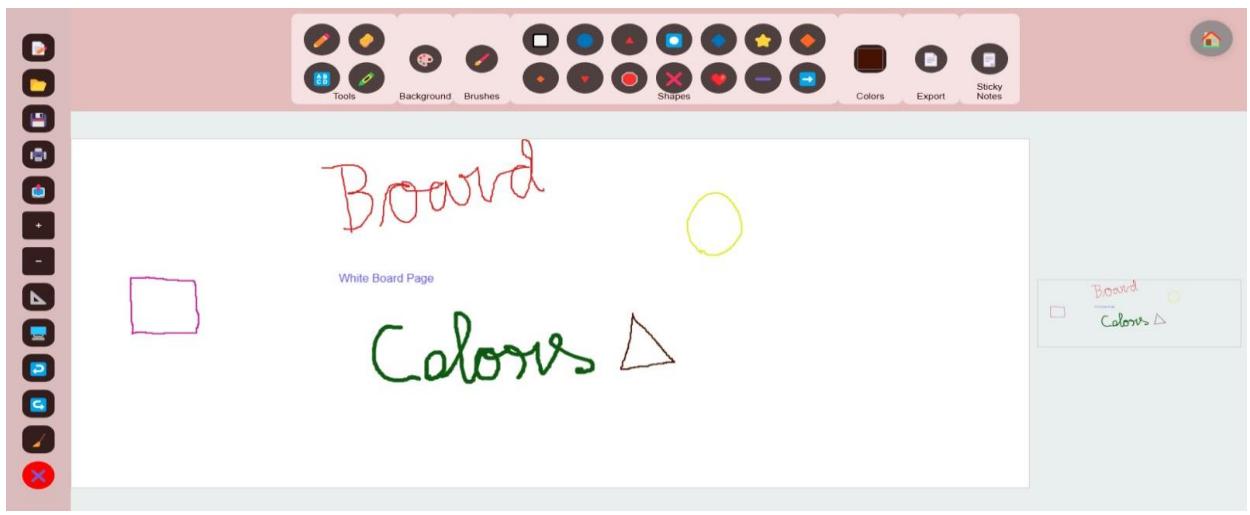
Brushes:



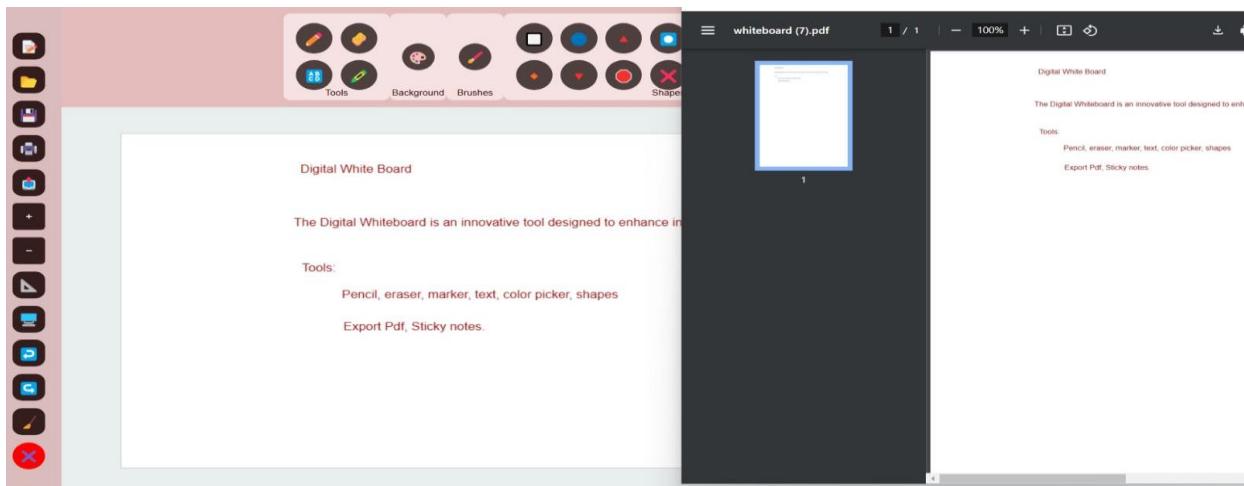
Shapes:



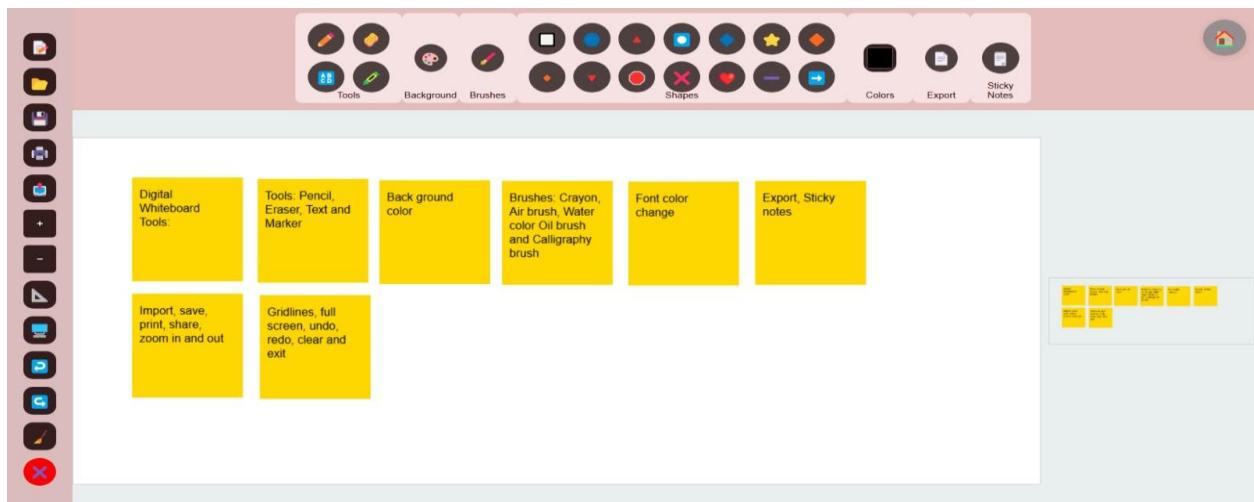
Colors:



Export:



Sticky Notes:



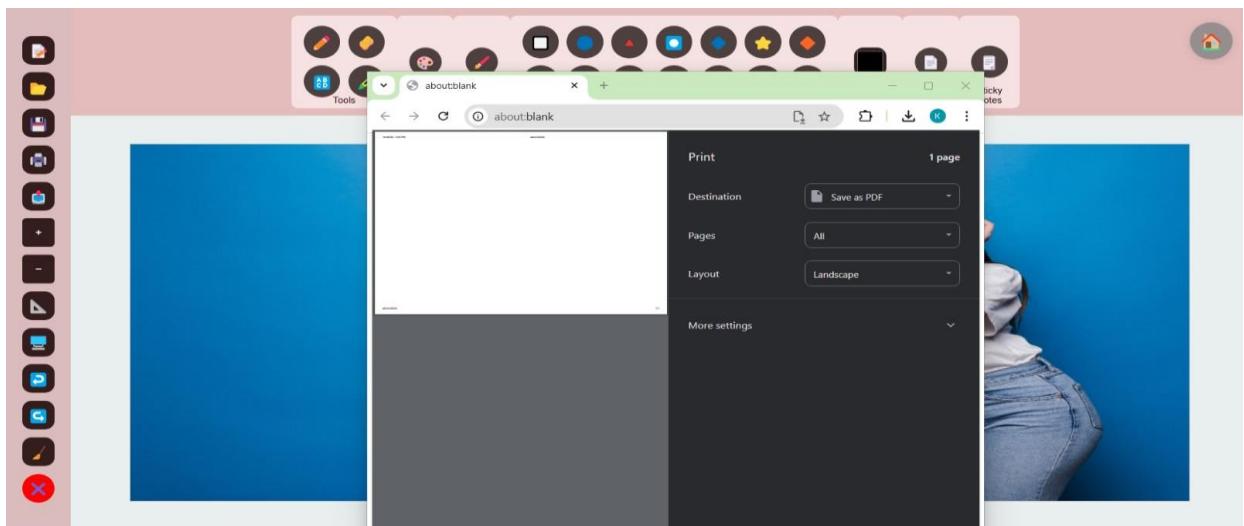
Import Image:



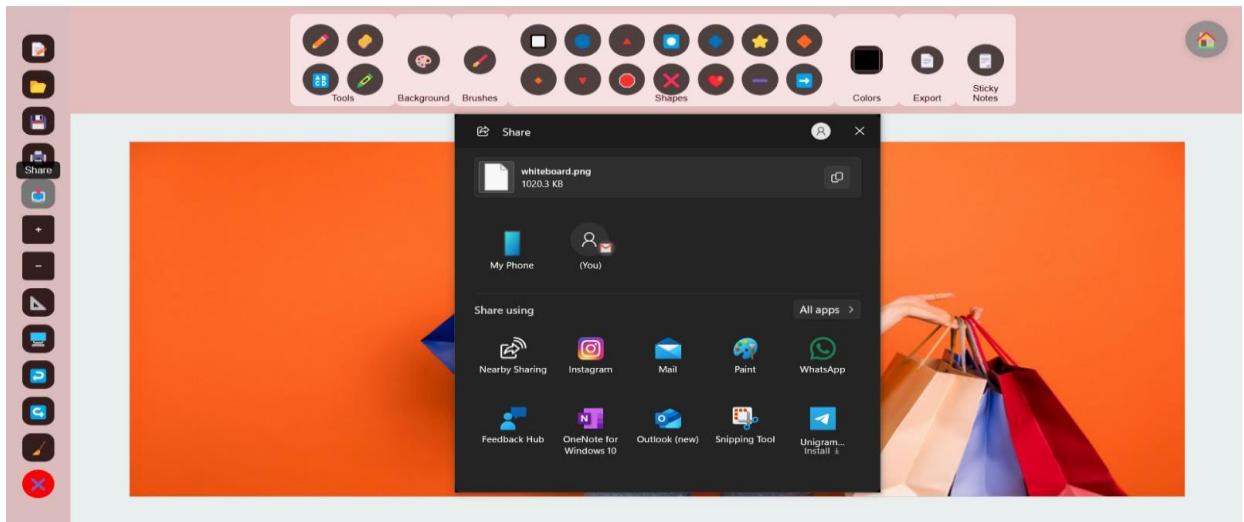
Save:



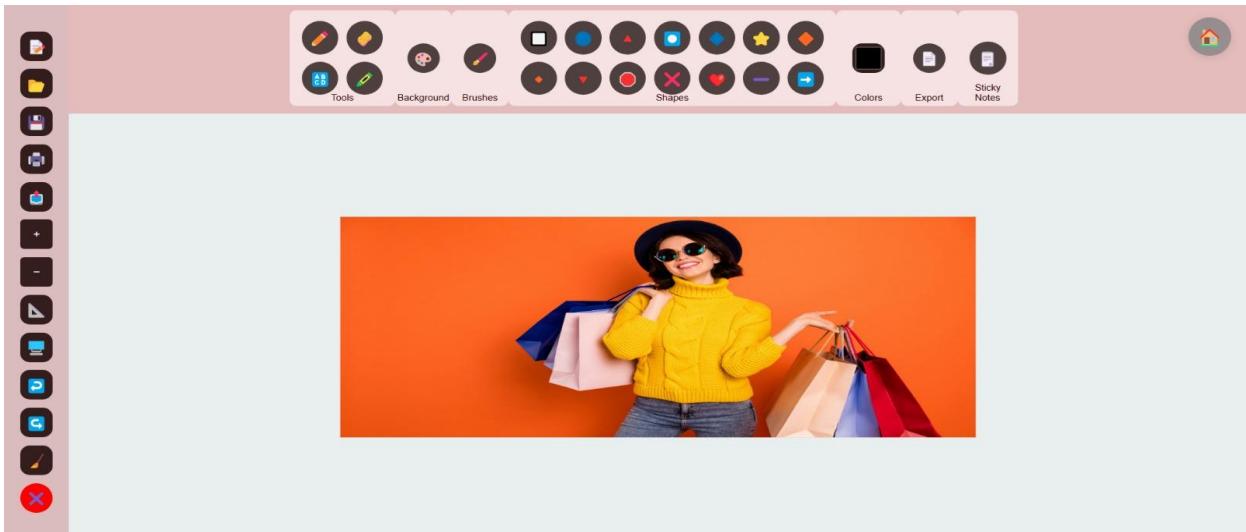
Print:



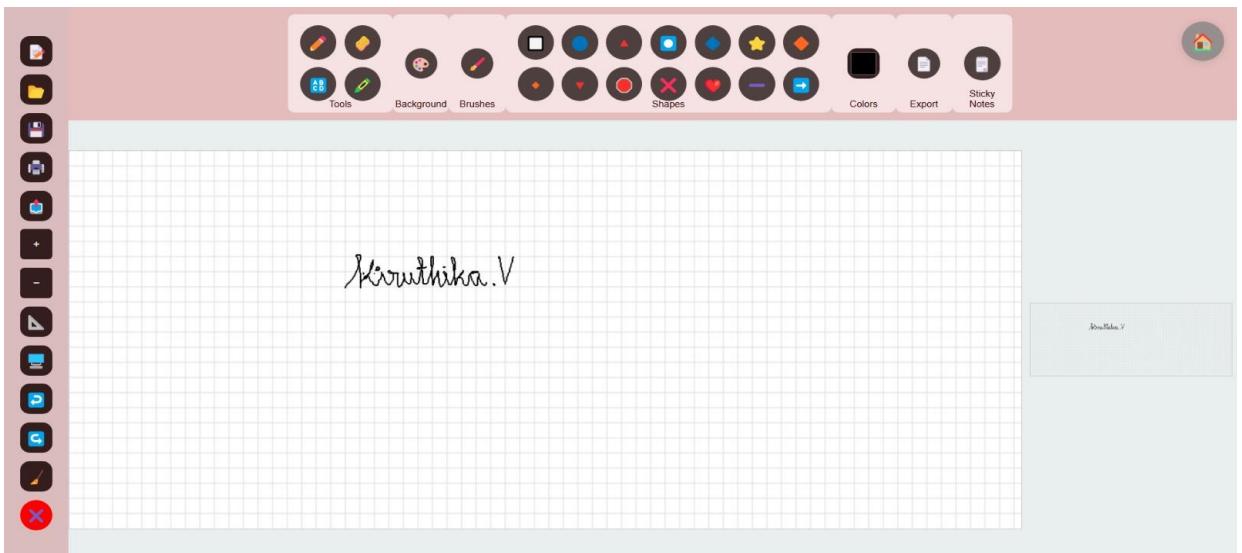
Share:



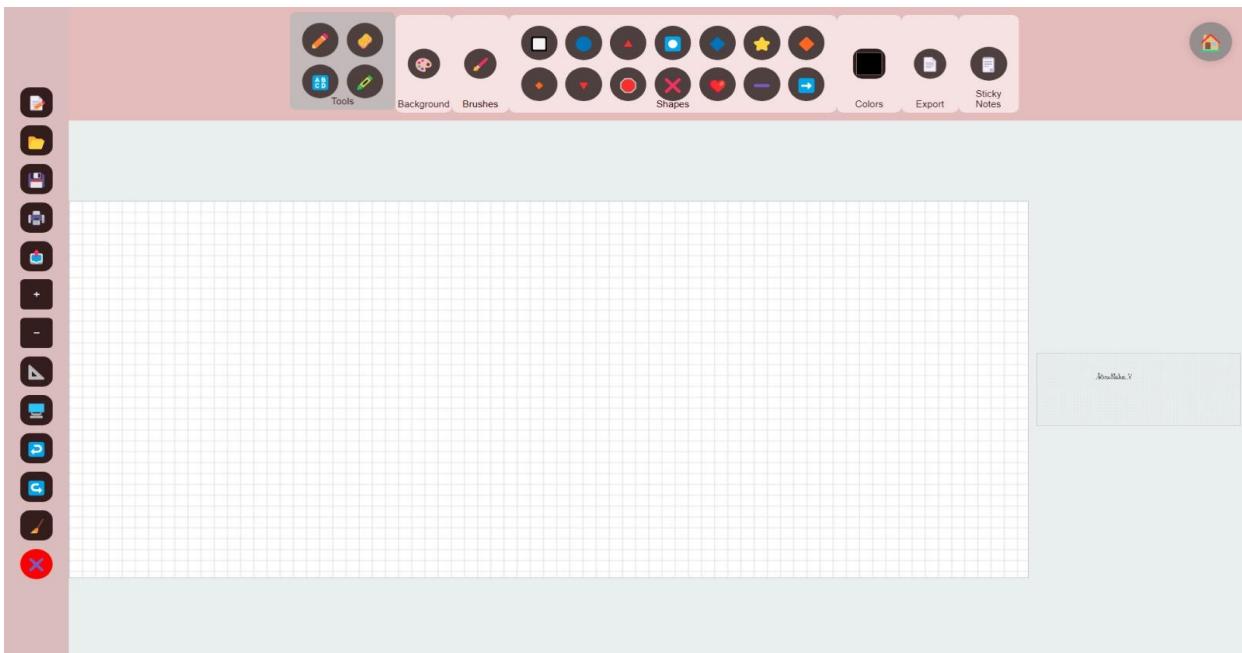
Zoom in and Zoom out:



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:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Front Page</title>
7      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
8      <link rel="stylesheet" href="styles.css">
9  </head>
10 <body>
11     <div class="slider">
12         <div class="slide" id="slide1">
13             <div class="header">
14                 <span class="name"> BlanqCanvas </span>
15                 <i class="fas fa-user login-icon" onclick="openLoginPopup()"></i>
16             </div>
17             <div class="welcome-message">
18                 <h1>Welcome to the Whiteboard</h1>
19                 <p>Unlock your creativity with the dynamic features of our Digital Whiteboard.
20                     | <br> Seamless, intuitive, and built for limitless possibilities.</p>
21                 <p>Click below to start using the whiteboard:</p>
22                 <button><a href="whiteboard.html" class="start-button">Go to Whiteboard</a></button>
23             </div>
24             <div class="navigation">
25                 <button class="prev" onclick="goToSlide(3)">&#8592; </button>
26                 <button class="next" onclick="goToSlide(2)">&#8594; </button>
27             </div>
28         </div>
29     <!-- Overlay -->
30     <div class="overlay" id="overlay" onclick="closeLoginPopup()"></div>
31     <!-- Login Popup -->
32     <div class="login-popup" id="loginPopup">
33         <h2>Login</h2>
34         <input type="text" id="username" placeholder="Enter Username">
35         <input type="password" id="password" placeholder="Enter Password">
36         <button onclick="login()">Login</button>
37         <button class="cancel-button" onclick="closeLoginPopup()">Cancel</button>
```

```
38      </div>
39      <div class="slide" id="slide2">
40          <div class="header">
41              <span class="name"> BlanqCanvas </span>
42              <i class="fas fa-user login-icon" onclick="openLoginPopup()"></i>
43          </div>
44          <h2>About</h2>
45          <p class="definition">A feature-rich digital whiteboard with tools for drawing, erasing, shaping, and customizing colors.<br>
46              Includes options for exporting, sticky notes, zoom, undo/redo, and gridlines for enhanced creativity. <br>
47              The design is user-friendly, enabling seamless creation, editing, and sharing of visual ideas. <br>
48              The layout is responsive and organized to enhance productivity and creativity.
49          </p>
50          <button onclick="goToSlide(3)">Next</button>
51          <div class="navigation">
52              <button class="prev" onclick="goToSlide(1)"> &#8592; </button>
53              <button class="next" onclick="goToSlide(3)"> &#8594; </button>
54          </div>
55      </div>
56      <div class="slide" id="slide3">
57          <div class="header">
58              <span class="name"> BlanqCanvas </span>
59              <i class="fas fa-user login-icon" onclick="openLoginPopup()"></i>
60          </div>
61          <h2>Features</h2>
62          <div class="features">
63              <button class="feature-btn" onclick="showFeature(this, 'Drawing and Erasing Tools')">
64                  <div class="front">Drawing and Erasing Tools</div>
65                  <div class="back">Intuitive tools like pencil, marker, and eraser for effortless sketching and editing.
66                  </div>
67              </button>
68              <button class="feature-btn" onclick="showFeature(this, 'Shape Creation')">
69                  <div class="front">Shape Creation</div>
70                  <div class="back">Predefined shapes such as squares, circles, arrows, and stars for structured diagrams.
```

```
71          </div>
72      </button>
73      <button class="feature-btn" onclick="showFeature(this, 'Brush Customization')">
74          <div class="front">Brush Customization</div>
75          <div class="back">Multiple brush styles, including watercolor, crayon, and calligraphy options.</div>
76      </button>
77      <button class="feature-btn" onclick="showFeature(this, 'Color Palette')">
78          <div class="front">Color Palette</div>
79          <div class="back">A color picker for selecting and applying your preferred brush or background color.</div>
80      </button>
81      <button class="feature-btn" onclick="showFeature(this, 'Sticky Notes')">
82          <div class="front">Sticky Notes</div>
83          <div class="back">Add and manage sticky notes for brainstorming or jotting down quick ideas.
84          </div>
85      </button>
86      <button class="feature-btn" onclick="showFeature(this, 'Export and Save Options')">
87          <div class="front">Export and Save Options</div>
88          <div class="back">Export your work as a PDF or save files for future use.
89          </div>
90      </button>
91      <button class="feature-btn" onclick="showFeature(this, 'Undo, Redo, and Clear')">
92          <div class="front">Undo, Redo, and Clear</div>
93          <div class="back">Simple controls to fix mistakes or clear the canvas for a fresh start.</div>
94      </button>
95      <button class="feature-btn" onclick="showFeature(this, 'Zoom and Gridlines')">
96          <div class="front">Zoom and Gridlines</div>
97          <div class="back">Zoom in/out and toggle gridlines for precise alignment and focus.</div>
98      </button>
99      <button class="feature-btn" onclick="showFeature(this, 'Fullscreen Mode')">
100          <div class="front">Fullscreen Mode</div>
101          <div class="back">Switch to fullscreen for an immersive whiteboard experience.
102          </div>
```

```
103          </button>
104      </div>
105      <div class="navigation">
106          <button class="prev" onclick="goToSlide(2)"> &#8592; </button>
107          <button class="next" onclick="goToSlide(1)"> &#8594; </button>
108      </div>
109  </div>
110  <script src="script.js"></script>
111 </body>
112 </html>
113
114
```

Styles.css:

```
1  * {
2    margin: 0;
3    padding: 0;
4    box-sizing: border-box;
5  }
6  body {
7    font-family: Arial, sans-serif;
8    margin: 0;
9    padding: 0;
10   display: flex;
11   justify-content: center;
12   align-items: center;
13   min-height: 100vh;
14   background-color: #f4f4f9;
15 }
16 /* Slider styles */
17 .slider {
18   width: 100%;
19   height: 100vh;
20   display: flex;
21   align-items: center;
22   justify-content: center;
23   background-color: #fadbd8 ;
24   background-image: linear-gradient(to bottom right ,#e6b0aa , #cidbe3 );
25   background-size:contain;
26   background-position: center;
27   position: relative;
28 }
29 /* Slide styles */
30 .slide {
31   display: flex;
32   flex-direction: column;
33   justify-content: center;
34   align-items: center;
35   text-align: center;
36   color: #b67c7c;
37   width: 100%;
```

```
38   height: 100%;
39   animation: fadeIn 1s ease-in-out;
40 }
41 /* Header section */
42 .header {
43   position: absolute;
44   top: 20px;
45   left: 20px;
46   display: flex;
47   justify-content: space-between;
48   width: 90%;
49   padding: 10px 0;
50 }
51 .name {
52   font-size: 3rem;
53   font-weight: bold;
54   margin-right: 10px;
55   color: #333;
56   font-style:italic;
57   animation: slideInLeft 1s ease-in-out;
58 }
59 /* Login */
60 .login-icon {
61   font-size: 24px;
62   color: #333;
63   width: 40px;
64   cursor: pointer;
65 }
66 .login-popup, .overlay {
67   display: none;
68 }
69 .overlay {
70   position: fixed;
71   top: 0;
72   left: 0;
73   width: 100%;
```

```
74   height: 100%;
75   background: rgba(0, 0, 0, 0.5);
76   z-index: 999;
77 }
78 .login-popup {
79   position: fixed;
80   top: 50%;
81   left: 50%;
82   transform: translate(-50%, -50%);
83   background: white;
84   padding: 30px;
85   box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
86   z-index: 1000;
87   width: 300px;
88   border-radius: 8px;
89   font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
90 }
91 .login-popup h2 {
92   margin-top: 0;
93   text-align: center;
94 }
95 .login-popup input {
96   width: calc(100% - 20px);
97   padding: 8px;
98   margin: 10px 0;
99   font-size: 16px;
100  border: 1px solid #ccc;
101  border-radius: 4px;
102 }
103 .login-popup button {
104   width: 100%;
105   padding: 10px;
106   font-size: 16px;
107   color: white;
108   background-color: #4CAF50;
109   border: none;
```

```
110   margin: 10px 0;
111   border-radius: 4px;
112   cursor: pointer;
113 }
114 .login-popup button:hover {
115   background-color: #28c2e9;
116 }
117 .login-popup .cancel-button {
118   background-color: #ff4336;
119 }
120 .login-popup .cancel-button:hover {
121   background-color: #e62980;
122 }
123 /* welcome message */
124 .welcome-message h1 {
125   font-size: 4rem;
126   margin-bottom: 20px;
127   color: #34495e ;
128   animation: fadeInUp 1.5s ease-out;
129 }
130 .welcome-message p {
131   font-size: 1.5rem;
132   font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif, Cochin, Georgia, Times, 'Times New Roman', serif;
133   margin-bottom: 20px;
134   color: #646464 ;
135   animation: fadeInUp 1.5s ease-out;
136 }
137 button {
138   padding: 12px 25px;
139   font-size: 18px;
140   background-color: #2196F3;
141   color: white;
142   border: none;
143   border-radius: 5px;
144   cursor: pointer;
145   margin-top: 10px;
```

```

146     transition: background-color 0.3s ease, transform 0.3s ease;
147   }
148   button a{
149     text-decoration: none;
150   }
151   button:hover {
152     background-color: #1b8c28;
153     transform: scale(1.1);
154   }
155   .prev, .next {
156     padding: 10px 20px;
157     font-size: 18px;
158     background-color: #rgba(0, 0, 0, 0.5);
159     color: white;
160     border: none;
161     border-radius: 5px;
162     cursor: pointer;
163     transition: background-color 0.3s ease, transform 0.3s ease;
164   }
165   .prev:hover, .next:hover {
166     background-color: #rgba(66, 59, 59, 0.7);
167     transform: scale(1.1);
168   }
169   #slide1 {
170     display: flex;
171     flex-direction: column;
172   }
173   #slide2 h2, #slide3 h2 {
174     font-size: 3rem;
175     margin-bottom: 20px;
176     color: #51383f;
177   }
178   #slide2 p{
179     font-size: 1.3rem;
180     color: #68182f;
181   }

```

```

182  /* Features */
183  .features {
184    display: grid;
185    grid-template-columns: repeat(3, 1fr);
186    gap: 15px;
187  }
188  .feature-btn {
189    position: relative;
190    width: 400px;
191    padding: 70px;
192    background-color: #007bff;
193    color: #fff;
194    border: none;
195    border-radius: 10px;
196    cursor: pointer;
197    transform: translateY(0);
198    transition: transform 0.5s ease, background-color 0.3s ease;
199    perspective: 1000px;
200    font-size: 16px;
201  }
202  .feature-btn:hover {
203    transform: translateY(-10px);
204    background-color: #0056b3;
205  }
206  .feature-btn.active .back {
207    transform: rotateY(0);
208  }
209  .feature-btn .front, .feature-btn .back {
210    position: absolute;
211    top: 0;
212    left: 0;
213    width: 100%;
214    height: 100%;
215    backface-visibility: hidden;
216    border-radius: 10px;
217    display: flex;
218  }

```

```

219   justify-content: center;
220   align-items: center;
221   font-size: 14px;
222 }
223 .feature-btn .front{
224   animation: slideInLeft 1s ease-in-out;
225 }
226 .feature-btn .back {
227   background-color: #6f6565;
228   color: #f4eaea;
229   transform: rotateY(180deg);
230 }
231 .feature-btn.active .front {
232   transform: rotateY(180deg);
233 }
234 .feature-btn.active .back {
235   transform: rotateY(0);
236 }
237 /* Navigation buttons */
238 .navigation {
239   position: absolute;
240   bottom: 20px;
241   display: flex;
242   justify-content: space-between;
243   width: 90%;
244   animation: fadeInUp 1s ease-in-out;
245 }
246
247 .navigation button {
248   padding: 10px 20px;
249   background-color: #007bff;
250   color: #fff;
251   border: none;
252   border-radius: 5px;
253   cursor: pointer;
254   margin: 5px;

```

```

255 }
256 .navigation button:hover {
257   background-color: #0056b3;
258 }

```

```

259 /* Media queries for responsiveness */
260 @media (max-width: 1250px) {
261   .feature-btn {
262     width: 300px;
263     padding: 50px;
264   }
265   .features {
266     display: grid;
267     grid-template-columns: repeat(3, 1fr);
268     gap: 15px;
269   }
270   .navigation button {
271     font-size: 14px;
272     padding: 8px 10px;
273   }
274   .name {
275     font-size: 3rem;
276   }
277 }
278 @media (max-width: 960px) {
279   .header {
280     flex-direction: column;
281     align-items: flex-start;
282   }
283   .login-icon {
284     width: 30px;
285   }
286   .name {
287     font-size: 2rem;
288   }
289   button {
290     font-size: 14px;
291     padding: 8px 16px;
292   }
293   .prev,
294   .next {
295     font-size: 14px;
296   }

```

```

297   padding: 8px ;
298 }
299 .feature-btn {
300   width: 250px;
301   padding: 60px;
302 }
303 .features {
304   display: grid;
305   grid-template-columns: repeat(2, 1fr);
306   gap: 10px;
307 }
308 @media (max-width: 540px) {
309   .welcome-message h1 {
310     font-size: 24px;
311   }
312   button {
313     font-size: 12px;
314     padding: 6px 12px;
315   }
316   .feature-btn {
317     width: 150px;
318     padding: 50px;
319   }
320   .features {
321     display: grid;
322     grid-template-columns: repeat(2, 1fr);
323     gap: 15px;
324   }
325   .prev,
326   .next {
327     font-size: 10px;
328     padding: 3px ;
329   }
330 }

```

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.

```
1  let currentSlide = 1;
2  function goToSlide(slideNumber) {
3      const slides = document.querySelectorAll('.slide');
4      slides.forEach((slide, index) => {
5          if (index === slideNumber - 1) {
6              slide.style.display = 'flex';
7          } else {
8              slide.style.display = 'none';
9          }
10     });
11     currentSlide = slideNumber;
12 }
13 function showFeature(feature) {
14     const featureInfo = document.getElementById('feature-info');
15     if (feature === 'feature1') {
16         featureInfo.innerHTML = "<p>Feature 1: This is the first feature description.</p>";
17     } else if (feature === 'feature2') {
18         featureInfo.innerHTML = "<p>Feature 2: This is the second feature description.</p>";
19     } else if (feature === 'feature3') {
20         featureInfo.innerHTML = "<p>Feature 3: This is the third feature description.</p>";
21     }
22 }
23 function showFeature(button, featureText) {
24     const allButtons = document.querySelectorAll('.feature-btn');
25     allButtons.forEach(btn => btn.classList.remove('active'));
26     button.classList.add('active');
27 }
28 function openLoginPopup() {
29     document.getElementById("loginPopup").style.display = "block";
30     document.getElementById("overlay").style.display = "block";
31 }
32 function closeLoginPopup() {
33     document.getElementById("loginPopup").style.display = "none";
34     document.getElementById("overlay").style.display = "none";
35 }
36 function login() {
37     const username = document.getElementById("username").value;
38     const password = document.getElementById("password").value;
39     if (username === "admin" && password === "12345") {
40         alert("Login Successful!");
41         closeLoginPopup();
42     } else {
43         alert("Invalid Username or Password");
44     }
45 }
46 // Initialize the first slide
47 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:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.4.0/jspdf.umd.min.js"></script>
7      <link rel="stylesheet" href="boardstyles.css">
8      <title>Digital WhiteBoard</title>
9  </head>
10 <body>
11     <!-- Top Row for containers -->
12     <div id="topRow">
13         <div class="container" id="toolsContainer">
14             <button class="iconBtn" id="pencilBtn" title="Pencil">铅笔 </button>
15             <button class="iconBtn" id="eraserBtn" title="Eraser">橡皮擦 </button>
16             <button class="iconBtn" id="textBtn" title="Text">文本 </button>
17             <button class="iconBtn" id="markerBtn" title="Marker">马克笔 </button>
18             <span>Tools</span>
19         </div>
20         <div class="container" id="backgroundColorContainer">
21             <button class="iconBtn" id="backgroundColorBtn" title="Background Color">背景颜色 </button>
22             <span>Background</span>
23         </div>
24         <div class="container" id="brushesContainer">
25             <button class="iconBtn" id="brushBtn" title="Brush">画笔 </button>
26             <div id="brushOptions" class="brush-options">
27                 <button class="iconBtn" id="crayonBtn" title="Crayon">蜡笔 </button>
28                 <button class="iconBtn" id="airbrushBtn" title="Air Brush">喷枪 </button>
29                 <button class="iconBtn" id="watercolorBtn" title="Watercolor">水彩 </button>
30                 <button class="iconBtn" id="oilBrushBtn" title="Oil Brush">油画笔 </button>
31                 <button class="iconBtn" id="calligraphyBtn" title="Calligraphy Brush">书法笔 </button>
32             </div>
33             <span>Brushes</span>
34         </div>
35         <div class="container" id="shapesContainer">
36             <button class="iconBtn" id="squareBtn" title="Square">正方形 </button>
37             <button class="iconBtn" id="circleBtn" title="Circle">圆形 </button>
```

```
38     <button class="iconBtn" id="triangleBtn" title="Triangle">▲</button>
39     <button class="iconBtn" id="ellipseBtn" title="Ellipse">●</button>
40     <button class="iconBtn" id="hexagonBtn" title="Hexagon">◆</button>
41     <button class="iconBtn" id="starBtn" title="Star">★</button>
42     <button class="iconBtn" id="pentagonBtn" title="Pentagon">◆</button>
43     <button class="iconBtn" id="rhombusBtn" title="Rhombus">◆</button>
44     <button class="iconBtn" id="parallelogramBtn" title="Parallelogram">▼</button>
45     <button class="iconBtn" id="octagonBtn" title="Octagon">●</button>
46     <button class="iconBtn" id="crossBtn" title="Cross">✗</button>
47     <button class="iconBtn" id="heartBtn" title="Heart">♥</button>
48     <button class="iconBtn" id="lineBtn" title="Line">■</button>
49     <button class="iconBtn" id="arrowBtn" title="Arrow">➡</button>
50   <span>Shapes</span>
51 </div>
52 <div class="container" id="colorsContainer">
53   <input class="tool" type="color" id="color-picker" title="Pick a color" onchange="changeBrushColor(this.value)">
54   <span>Colors</span>
55 </div>
56 <div class="container" id="exportContainer">
57   <button class="iconBtn" id="exportPdfBtn" title="Export PDF">PDF</button>
58   <span>Export</span>
59 </div>
60 <div id="stickyNotesContainer" class="container">
61   <button class="iconBtn" id="addStickyNoteBtn" title="Add Sticky Note">+</button>
62   <span>Sticky Notes</span>
63 </div>
64 </div>
65 <!-- Left Sidebar for tools -->
66 <div id="toolbar">
67   <div class="tool-container">
68     <button class="tool" data-name="New" id="newBtn">New</button>
69   </div>
70   <div class="tool-container">
```

```
71     <button class="tool" data-name="Import" id="importBtn">Import</button>
72   </div>
73   <div class="tool-container">
74     <button class="tool" data-name="Save As" id="saveBtn">Save As</button>
75   </div>
76   <div class="tool-container">
77     <button class="tool" data-name="Print" id="printBtn">Print</button>
78   </div>
79   <div class="tool-container">
80     <button class="tool" data-name="Share" id="shareBtn">Share</button>
81   </div>
82   <div class="tool-container">
83     <button class="tool zoom-btn" data-name="Zoom In" id="zoomInBtn">+</button>
84   </div>
85   <div class="tool-container">
86     <button class="tool zoom-btn" data-name="Zoom Out" id="zoomOutBtn">-</button>
87   </div>
88   <div class="tool-container">
89     <button class="tool" data-name="Gridlines" id="gridBtn">Gridlines</button>
90   </div>
91   <div class="tool-container">
92     <button class="tool" data-name="Full Screen" id="fullscreenBtn">Fullscreen</button>
93   </div>
94   <div class="tool-container">
95     <button class="tool" data-name="Undo" id="undoBtn">Undo</button>
96   </div>
97   <div class="tool-container">
98     <button class="tool" data-name="Redo" id="redoBtn">Redo</button>
99   </div>
100  <div class="tool-container">
101    <button class="tool" data-name="Clear" id="clearBtn">Clear</button>
102  </div>
103  <div class="tool-container">
```

```

104     |     <button class="tool exit-btn" data-name="Exit" id="exitBtn">X</button>
105     |   </div>
106   </div>
107   |   <!-- Go to Home Button --&gt;
108   |   &lt;a href="index.html" id="goHomeBtn" title="Go to Home"&gt;H&lt;/a&gt;
109   |   <!-- Main Content Area --&gt;
110   |   &lt;div id="content"&gt;
111     |     <!-- Whiteboard --&gt;
112     |     &lt;div id="whiteboard-wrapper"&gt;
113       |       &lt;canvas id="whiteboard"&gt;&lt;/canvas&gt;
114     |     &lt;/div&gt;
115     |     <!-- Thumbnail --&gt;
116     |     &lt;div&gt;
117       |       &lt;img id="thumbnailPreview" alt="Thumbnail Preview" style="display: none; margin: 10px; border: 1px solid #ccc;" /&gt;
118     |     &lt;/div&gt;
119   &lt;/div&gt;
120   |   &lt;script src="boardscript.js"&gt;&lt;/script&gt;
121 &lt;/body&gt;
122 &lt;/html&gt;
</pre>

```

Boardstyle.css:

```

1  body {
2   margin: 0;
3   display: flex;
4   flex-direction: column;
5   height: 100vh;
6   background-color: #e9e9e9;
7   font-family: Arial, sans-serif;
8 }
9 /* Top Row */
10 #topRow {
11   display: flex;
12   justify-content: center;
13   padding: 10px;
14   background-color: #e5bcbe;
15   margin-left: 70px;
16   flex-direction: row;
17 }
18 /* Main Container */
19 .container {
20   display: flex;
21   flex-direction: row;
22   align-items: center;
23   justify-content: center;
24   gap: 20px;
25   background-color: #f6e2e2;
26   border-radius: 8px;
27   padding: 15px;
28   color: #rgb(242, 86, 86);
29   position: relative;
30   transition: background-color 0.3s ease, transform 0.3s ease;
31 }
32 .container:hover {
33   background-color: #aca9a9;
34   transform: translateY(-5px);
35 }
36 .container span {
37   position: absolute;
38   bottom: 5px;
39   left: 50%;
40   transform: translateX(-50%);
41   color: #rgb(43, 2, 2);
42   font-size: 12px;
43   padding: 0 5px;
44   transition: color 0.3s ease;
45 }
46 /* Icon Button */
47 .iconBtn {
48   background-color: #4f4040;
49   border: none;
50   color: white;
51   font-size: 20px;
52   width: 40px;
53   height: 40px;
54   display: flex;
55   align-items: center;
56   justify-content: center;
57   border-radius: 50%;
58   cursor: pointer;
59   position: relative;
60   transition: transform 0.3s ease, background-color 0.3s ease;
61 }
62 .iconBtn:hover {
63   background-color: #777;
64   border-radius: 5px;
65   transform: scale(1.1);
66 }
67 .iconBtn::after {
68   content: attr(title);
69   position: absolute;
70   bottom: 100%;
71   left: 50%;
72   transform: translateX(-50%);
73   background-color: #rgba(0, 0, 0, 0.8);

```

```

74  color: ■#fff;
75  font-size: 12px;
76  padding: 5px 10px;
77  border-radius: 5px;
78  white-space: nowrap;
79  max-width: 120px;
80  text-align: center;
81  visibility: hidden;
82  opacity: 0;
83  transition: visibility 0.2s, opacity 0.2s;
84  z-index: 10;
85 }
86 .iconBtn:hover::after {
87  visibility: visible;
88  opacity: 1;
89 }
90 /* Tool containers */
91 #toolsContainer, #shapesContainer, #colorsContainer, #stickyNotesContainer {
92  display: flex;
93  flex-direction: row;
94  gap: 5px;
95  justify-content: center;
96 }
97 #toolsContainer .iconBtn,
98 #shapesContainer .iconBtn,
99 #colorsContainer .iconBtn,
100 #stickyNotesContainer .iconBtn {
101 width: 45px;
102 height: 45px;
103 }
104 #backgroundColorContainer,
105 #importContainer {
106  display: flex;
107  align-items: center;
108 }
109 #backgroundColorBtn {

```

```

110  font-size: 20px;
111  padding: 5px;
112 }
113 #importPdfBtn {
114  font-size: 20px;
115  padding: 5px;
116 }
117 /* Whiteboard Section */
118 #content {
119  display: flex;
120  justify-content: center;
121  align-items: center;
122  flex: 1;
123  margin-left: 80px;
124 }
125 #whiteboard-wrapper {
126  display: flex;
127  justify-content: center;
128  align-items: center;
129  height: 80%;
130  width: 90%;
131 }
132 #whiteboard {
133  width: 100%;
134  height: 500px;
135  background-color: ■#fff;
136  cursor: crosshair;
137  border: 1px solid ■#ccc;
138  box-sizing: border-box;
139 }
140 /* Go Home Button */
141 #goHomeBtn {
142  position: fixed;
143  top: 20px;
144  right: 20px;
145  background-color: ■#978f8f;

```

```

146  padding: 10px;
147  border-radius: 50%;
148  font-size: 24px;
149  text-decoration: none;
150  color: □#333;
151  box-shadow: 0px 2px 10px □rgba(0, 0, 0, 0.1);
152 }
153 #goHomeBtn:hover {
154  background-color: ■#e0e0e0;
155  cursor: pointer;
156  transform: scale(1.05);
157 }
158 #toolsContainer {
159  display: grid;
160  grid-template-columns: repeat(2, 1fr);
161  gap: 10px;
162 }
163 #shapesContainer {
164  display: grid;
165  grid-template-columns: repeat(7, 1fr);
166  gap: 10px;
167 }
168 /* Toolbar */
169 #toolbar {
170  width: 70px;
171  background-color: ■#dbbcbe;
172  color: ■#fff;
173  display: flex;
174  flex-direction: column;
175  justify-content: center;
176  padding: 5px;
177  gap: 1px;
178  position: fixed;
179  top: 0;
180  bottom: 0;
181  z-index: 10;

```

```

182 }
183 /* Tool Container */
184 .tool-container {
185  display: flex;
186  flex-direction: column;
187  align-items: center;
188  position: relative;
189 }
190 .tool {
191  width: 40px;
192  height: 40px;
193  display: flex;
194  align-items: center;
195  justify-content: center;
196  background-color: □#341d1d;
197  border: none;
198  border-radius: 30%;
199  cursor: pointer;
200  color: ■white;
201  font-size: 20px;
202  margin: 5px;
203  position: relative;
204  transition: cubic-bezier(0.075, 0.82, 0.165, 1);
205 }
206 .tool:hover {
207  background-color: ■#777;
208  transform: scale(1.05);
209 }
210 .tool.active {
211  background-color: ■#6e6b6b;
212 }
213 /* Tool name shown on hover */
214 .tool:hover::after {
215  content: attr(data-name);
216  position: absolute;
217  bottom: 100%;
```

```

218     left: 50%;  

219     transform: translateX(-50%);  

220     background-color: □rgba(0, 0, 0, 0.8);  

221     color: ■#fff;  

222     font-size: 12px;  

223     padding: 5px 10px;  

224     border-radius: 5px;  

225     white-space: nowrap;  

226     max-width: 120px;  

227     text-align: center;  

228     z-index: 20;  

229   }  

230   /* Zoom Buttons */  

231   .zoom-btn {  

232     background-color: □#341d1d;  

233     border-radius: 5px;  

234     color: ■white;  

235     padding: 8px 12px;  

236     cursor: pointer;  

237     font-size: 14px;  

238   }  

239  

240   .zoom-btn:hover {  

241     background-color: ■#777;  

242     transform: scale(1.05);  

243   }  

244   /* Exit Button */  

245   .exit-btn {  

246     background-color: ■red;  

247     color: ■white;  

248     font-size: 18px;  

249     border-radius: 50%;  

250     padding: 10px;  

251     cursor: pointer;  

252   }  

253   .exit-btn:hover {  


```

```

254     background-color: ■darkred;  

255     transform: scale(1.05);  

256   }  

257   /* Brush Options */  

258   #brushOptions {  

259     display: none;  

260     position: absolute;  

261     top: 100%;  

262     left: 50%;  

263     transform: translateX(-50%);  

264     background-color: ■#fff;  

265     box-shadow: 0 4px 8px □rgba(0, 0, 0, 0.2);  

266     border-radius: 8px;  

267     padding: 10px;  

268     z-index: 100;  

269   }  

270   #brushesContainer {  

271     position: relative;  

272   }  

273   #brushesContainer:hover #brushOptions {  

274     display: block;  

275     transform: scale(1.05);  

276   }  

277   #brushOptions button {  

278     margin: 5px;  

279   }  

280   #brushOptions button:hover::after {  

281     content: attr(title);  

282     position: absolute;  

283     bottom: 30px;  

284     left: 50%;  

285     transform: translateX(-50%);  

286     background-color: □#000;  

287     color: ■#fff;  

288     padding: 2px 5px;  

289     border-radius: 3px;  


```

```

290     font-size: 14px;  

291     white-space: nowrap;  

292   }  

293   /* Sticky Notes */  

294   .sticky-note {  

295     position: absolute;  

296     background-color: ■#ffeb3b;  

297     padding: 10px;  

298     border-radius: 8px;  

299     width: 150px;  

300     height: 150px;  

301     box-shadow: 0px 4px 10px □rgba(0, 0, 0, 0.1);  

302     cursor: move;  

303   }  

304   .sticky-note .note-text {  

305     min-height: 100%;  

306     white-space: pre-wrap;  

307     word-wrap: break-word;  

308     font-family: Arial, sans-serif;  

309   }  

310   .sticky-note .delete-btn {  

311     position: absolute;  

312     top: 5px;  

313     right: 5px;  

314     background: none;  

315     border: none;  

316     font-size: 20px;  

317     cursor: pointer;  

318   }  

319   .sticky-note .resize-handle {  

320     position: absolute;  

321     bottom: 5px;  

322     right: 5px;  

323     width: 15px;  

324     height: 15px;  

325     background: □#000;  


```

```

326     cursor: se-resize;  

327   }  

328   .sticky-note.locked {  

329     pointer-events: none;  

330   }  

331   .sticky-note.dragging {  

332     opacity: 0.5;  

333   }  

334   #thumbnailPreview {  

335     max-width: 250px;  

336     max-height: 250px;  

337   }  

338 }
```

```

341 @media (max-width: 1000px) {
342   #topRow {
343     flex-direction: column;
344     display: grid;
345     grid-template-columns: repeat(3, 1fr);
346     gap: 10px;
347   }
348   #toolsContainer, #shapesContainer {
349     flex-direction: column;
350     display: grid;
351     justify-content: center;
352     align-items: center;
353     grid-template-columns: repeat(3, 1fr);
354     gap: 5px;
355   }
356   #toolbar {
357     width: 60px;
358   }
359   .iconBtn {
360     width: 35px;
361     height: 35px;
362   }
363   .tool {
364     width: 35px;
365     height: 35px;
366   }
367   #whiteboard {
368     height: 400px;
369   }
370   #brushOptions {
371     grid-template-columns: repeat(1, 1fr);
372     padding: 4px;
373     font-size: 10px;
374     position: relative;
375     top: 0;
376     left: 0;

```

```

377   width: 100%;
378 }
379 }
380 }
381 }
382 @media (max-width: 480px) {
383   /* Containers */
384   #toolsContainer, #shapesContainer {
385     flex-direction: column;
386     display: grid;
387     justify-content: center;
388     align-items: center;
389     grid-template-columns: repeat(2, 1fr);
390     gap: 5px;
391   }
392   #content {
393     flex-direction: column;
394     margin-left: 0;
395     display: flex;
396     align-items: center;
397     justify-content: center;
398   }
399   #whiteboard-wrapper {
400     width: 100%;
401     height: 70%;
402   }
403   #whiteboard {
404     height: 300px;
405   }
406   /* Icon Button */
407   .iconBtn {
408     background-color: □#4f4040;
409     border: none;
410     color: ■white;
411     font-size: 20px;

```

```

413   height: 40px;
414   display: flex;
415   align-items: center;
416   justify-content: center;
417   border-radius: 50%;
418   cursor: pointer;
419   position: relative;
420   transition: transform 0.3s ease, background-color 0.3s ease;
421 }
422 /* Tool Button */
423 .tool {
424   width: 40px;
425   height: 40px;
426   display: flex;
427   align-items: center;
428   justify-content: center;
429   background-color: □#341d1d;
430   border: none;
431   border-radius: 30%;
432   cursor: pointer;
433   color: ■white;
434   font-size: 20px;
435   margin: 5px;
436   position: relative;
437   transition: transform 0.3s ease, cubic-bezier(0.075, 0.82, 0.165, 1);
438 }
439 #brushOptions {
440   position: relative;
441   top: 0;
442   left: 0;
443   width: 100%;
444   transform: none;
445   padding: 5px;
446   font-size: 14px;
447 }

```

```

384 /* Animation effect */
385 @keyframes hoverEffect {
386   0% {
387     background-color: □#f6e2e2;
388   }
389   100% {
390     background-color: □#aca9a9;
391   }
392 }
393 .container:hover {
394   animation: hoverEffect 0.3s forwards;
395 }
396 @keyframes iconHoverEffect {

```

```

397   0% {
398     transform: scale(1);
399     background-color: #4f4040;
400   }
401   50% {
402     transform: scale(1.2);
403     background-color: #3d3232;
404   }
405   100% {
406     transform: scale(1);
407     background-color: #4f4040;
408   }
409 }
410 .iconBtn:hover, .tool:hover, #goHomeBtn:hover {
411   animation: iconHoverEffect 0.4s ease-in-out;
412 }
413 @keyframes fadeIn {
414   0% {
415     opacity: 0;
416     transform: translateY(-10px);
417   }
418   100% {
419     opacity: 1;
420     transform: translateY(0);
421   }
422 }
423 #toolbar, #topRow .iconBtn{
424   animation: fadeIn 0.5s ease-out;
425 }
426 @keyframes whiteboardAnimation {
427   0% {
428     background-color: #f9f9f9;
429     border-color: #e0e0e0;
430     transform: scale(0.9);
431     opacity: 0.8;
432   }
433   50% {
434     background-color: #fff;
435     border-color: #ccc;
436     transform: scale(1.05);
437     opacity: 1;
438   }
439   100% {
440     background-color: #fff;
441     border-color: #ccc;
442     transform: scale(1);
443     opacity: 1;
444   }
445 }
446 #whiteboard {
447   animation: whiteboardAnimation 1.5s ease-in-out;
448 }

```

Boardscript.js:

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

```

1 // Ensure DOM is fully loaded before running the script
2 document.addEventListener("DOMContentLoaded", () => {
3   // Get elements for whiteboard tools
4   const newBtn = document.getElementById("newBtn");
5   const importBtn = document.getElementById("importBtn");
6   const saveBtn = document.getElementById("saveBtn");
7   const printBtn = document.getElementById("printBtn");
8   const shareBtn = document.getElementById("shareBtn");
9   const exitBtn = document.getElementById("exitBtn");
10  const zoomInBtn = document.getElementById("zoomInBtn");
11  const zoomOutBtn = document.getElementById("zoomOutBtn");
12  const gridBtn = document.getElementById("gridBtn");
13  const fullscreenBtn = document.getElementById("fullscreenBtn");
14  const undoBtn = document.getElementById("undoBtn");
15  const redoBtn = document.getElementById("redoBtn");
16
17  // Get canvas elements
18  const whiteboard = document.getElementById("whiteboard");
19  const ctx = whiteboard.getContext("2d");

```

```

446 // New canvas
447 newBtn.addEventListener("click", () => {
448   ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
449   lines = [];
450   undoneLines = [];
451   if (showGrid) drawGrid();
452 });
453
454 // Import image
455 importBtn.addEventListener("click", () => {
456   const input = document.createElement("input");
457   input.type = "file";
458   input.accept = "image/*";
459   input.addEventListener("change", (e) => {
460     const file = e.target.files[0];
461     const reader = new FileReader();
462     reader.onload = (e) => {
463       const img = new Image();
464       img.onload = () => {
465         ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
466         ctx.drawImage(img, 0, 0, whiteboard.width, whiteboard.height);
467         if (showGrid) drawGrid();
468       };
469       img.src = e.target.result;
470     };
471     reader.readAsDataURL(file);
472   });
473   input.click();
474 });
475
476 // Save functionality
477 saveBtn.addEventListener("click", async () => {
478   if (window.showSaveFilePicker) {
479     try {
480       const fileHandle = await window.showSaveFilePicker({

```

```

481   suggestedName: "whiteboard.png",
482   types: [
483     { description: "PNG Image", accept: { "image/png": [".png"] } }
484   ];
485 });
486 const writable = await fileHandle.createWritable();
487 const dataUrl = whiteboard.toDataURL("image/png");
488 const blob = await (await fetch(dataUrl)).blob();
489 await writable.write(blob);
490 await writable.close();
491 alert("File saved successfully!");
492 } catch (err) {
493   console.error("Save operation failed:", err);
494 }
495 } else {
496   alert("The File System Access API is not supported in this browser.");
497 }
498 });
499
500 // Print canvas content
501 printBtn.addEventListener("click", () => {
502   const dataUrl = whiteboard.toDataURL("image/png");
503   const printWindow = window.open();
504   printWindow.document.write(``);
505   printWindow.print();
506 });
507
508 // Share canvas content
509 shareBtn.addEventListener("click", () => {
510   const dataUrl = whiteboard.toDataURL("image/png");
511   navigator.share({ files: [new File([dataUrl], "whiteboard.png", { type: "image/png" })] });
512 });

```

```

// 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();

```

```

550   ctx.moveTo(0, y);
551   ctx.lineTo(whiteboard.width, y);
552   ctx.stroke();
553 }
554 }
555
556 // Redraw the canvas
557 function redrawCanvas() {
558   ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
559   if (showGrid) drawGrid();
560   lines.forEach((line) => {
561     ctx.beginPath();
562     ctx.moveTo(line.x1, line.y1);
563     ctx.lineTo(line.x2, line.y2);
564     ctx.strokeStyle = line.color;
565     ctx.lineWidth = line.width;
566     ctx.stroke();
567   });
568 }
569
570 // Start drawing
571 whiteboard.addEventListener("mousedown", (e) => {
572   drawing = true;
573   currentLine = {
574     startX: e.offsetX,
575     startY: e.offsetY,
576     color: "#fff",
577     width: 4,
578   };
579   undoneLines = []; // Clear redo history when starting a new drawing
580 });
581
582 // Stop drawing
583 whiteboard.addEventListener("mouseup", () => {
584   drawing = false;

```

```

584     drawing = false;
585     currentLine = null;
586   });
587
588   // Undo functionality
589   undoBtn.addEventListener("click", () => {
590     if (lines.length > 0) {
591       const lastLine = lines.pop();
592       undoneLines.push(lastLine); // Save the undone line for redo
593       redrawCanvas();
594     }
595   });
596
597   // Redo functionality
598   redoBtn.addEventListener("click", () => {
599     if (undoneLines.length > 0) {
600       const lastUndoneLine = undoneLines.pop();
601       lines.push(lastUndoneLine); // Restore the undone line
602       redrawCanvas();
603     }
604   });
605
606   // Toggle gridlines
607   gridBtn.addEventListener("click", () => {
608     showGrid = !showGrid;
609     redrawCanvas();
610   });
611
612   // Resize the canvas on window resize
613   window.addEventListener("resize", resizeCanvas);
614   resizeCanvas();

```

```

750  //clear functionality
751  const clearBtn = document.getElementById('clearBtn');
752  let isSelecting = false;
753  let selectedObject = null;
754
755  // Get references to the canvas and the thumbnail image
756  const thumbnailPreview = document.getElementById('thumbnailPreview');
757  // Function to capture the thumbnail
758  function captureThumbnail() {
759
760    if (whiteboard) {
761      const thumbnail = whiteboard.toDataURL('image/png');
762      // Set the thumbnail as the source for the preview image
763      thumbnailPreview.src = thumbnail;
764      thumbnailPreview.style.display = 'block'; // Make sure the thumbnail is visible
765    }
766  }
767
768  whiteboard.addEventListener('mouseup', captureThumbnail);
769  whiteboard.addEventListener('touchend', captureThumbnail);
770
771  // Function to clear the whiteboard
772  clearBtn.addEventListener('click', () => {
773    const ctx = whiteboard.getContext('2d');
774    ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
775  });
776

```

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

```

21   // Initialize canvas context
22   let zoomLevel = 1;
23   let lastX = 0;
24   let lastY = 0;
25   let lines = [];
26   let undoneLines = [];
27   let showGrid = false;
28
29   let currentTool = 'pencil'; // Track the selected tool
30   let drawing = false;
31   let isErasing = false;
32
33   // Tool Buttons
34   const pencilBtn = document.getElementById("pencilBtn");
35   const eraserBtn = document.getElementById("eraserBtn");
36   const textBtn = document.getElementById("textBtn");
37   const markerBtn = document.getElementById("markerBtn");
38
39   // Tool Selection Handlers
40   pencilBtn.addEventListener("click", () => {
41     currentTool = 'pencil';
42     isErasing = false;
43     ctx.strokeStyle = "#000";
44     ctx.lineWidth = 2;
45   });
46
47   let eraserSize = 50;
48
49   // Event listener for the eraser button
50   eraserBtn.addEventListener("click", () => {
51     currentTool = 'eraser';
52     ctx.globalCompositeOperation = 'destination-out'; // Set to eraser mode
53   });

```

```

55  // Erase function to clear an area
56  function erase(x, y) {
57    ctx.beginPath();
58    ctx.arc(x, y, eraserSize / 2, 0, Math.PI * 2); // Create a circular eraser
59    ctx.fill(); // Clear the area inside the circle
60  }
61
62 // Mouse events for erasing
63 whiteboard.addEventListener("mousemove", (e) => {
64   if (currentTool === 'eraser' && drawing) {
65     erase(e.offsetX, e.offsetY);
66   }
67 });
68
69 whiteboard.addEventListener("mouseup", () => {
70   if (currentTool === 'eraser') {
71     ctx.globalCompositeOperation = 'source-over'; // Reset to normal mode
72   }
73 });
74
75 markerBtn.addEventListener("click", () => {
76   currentTool = 'marker';
77   isErasing = false;
78   ctx.strokeStyle = "#FF0000";
79   ctx.lineWidth = 5;
80 });
81
82 // Event listener for the text tool button
83 textBtn.addEventListener("click", () => {
84   currentTool = 'text';
85 });
86
87 // Mouse Events for Drawing/Erasing/Text
88 whiteboard.addEventListener("mousedown", (e) => {

```

```

88
89   drawing = true;
90   lastX = e.offsetX;
91   lastY = e.offsetY;
92
93   if (currentTool === 'text') {
94     drawing = false; // Stop drawing when using the text tool
95     const text = prompt("Enter text:");
96     if (text) {
97       ctx.font = "16px Arial"; // Font size and style
98       ctx.fillStyle = ctx.strokeStyle; // Use current stroke style for text color
99       ctx.fillText(text, e.offsetX, e.offsetY); // Draw text at the clicked position
100    }
101  }
102 });
103
104 whiteboard.addEventListener("mousemove", (e) => {
105   if (!drawing || currentTool === 'text') return;
106
107   const currentX = e.offsetX;
108   const currentY = e.offsetY;
109
110   if (isErasing) {
111     erase(currentX, currentY); // Use existing erase function
112   } else {
113     ctx.beginPath();
114     ctx.moveTo(lastX, lastY);
115     ctx.lineTo(currentX, currentY);
116     ctx.stroke();
117
118     // Save the line for undo/redo functionality
119     lines.push({ x1: lastX, y1: lastY, x2: currentX, y2: currentY });
120   }
121
122   lastX = currentX;

```

```

123     lastY = currentY;
124   });
125
126   // Stop drawing when mouse is released
127   whiteboard.addEventListener("mouseup", () => {
128     drawing = false;
129   });
130
131   // Ensure the drawing stops when the mouse leaves the canvas
132   whiteboard.addEventListener("mouseout", () => {
133     drawing = false;
134   });
135
136   // Resize canvas
137   function resizeCanvas() {
138     whiteboard.width = window.innerWidth - 80;
139     whiteboard.height = window.innerHeight * 0.8;
140     if (showGrid) drawGrid(); // Redraw the grid after resizing
141   }
142   resizeCanvas();
143
144   // Clear canvas
145   function clearCanvas() {
146     ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
147   }
148

```

Java script code for shapes functionality:

```

149  // Store drawn shapes in an array
150  let shapes = [];
151  let currentShape = null;
152  let drawingShape = false;
153  let startX, startY;
154
155  // Event listeners for shape buttons
156  document.getElementById("squareBtn").addEventListener("click", () => setCurrentShape("square"));
157  document.getElementById("circleBtn").addEventListener("click", () => setCurrentShape("circle"));
158  document.getElementById("triangleBtn").addEventListener("click", () => setCurrentShape("triangle"));
159  document.getElementById("ellipseBtn").addEventListener("click", () => setCurrentShape("ellipse"));
160  document.getElementById("hexagonBtn").addEventListener("click", () => setCurrentShape("hexagon"));
161  document.getElementById("starBtn").addEventListener("click", () => setCurrentShape("star"));
162  document.getElementById("pentagonBtn").addEventListener("click", () => setCurrentShape("pentagon"));
163  document.getElementById("rhombusBtn").addEventListener("click", () => setCurrentShape("rhombus"));
164  document.getElementById("parallelogramBtn").addEventListener("click", () => setCurrentShape("parallelogram"));
165  document.getElementById("octagonBtn").addEventListener("click", () => setCurrentShape("octagon"));
166  document.getElementById("crossBtn").addEventListener("click", () => setCurrentShape("cross"));
167  document.getElementById("heartBtn").addEventListener("click", () => setCurrentShape("heart"));
168  document.getElementById("lineBtn").addEventListener("click", () => setCurrentShape("line"));
169  document.getElementById("arrowBtn").addEventListener("click", () => setCurrentShape("arrow"));
170
171  // Function to set the current shape to draw
172  function setCurrentShape(shape) {
173    currentShape = shape;
174    drawingShape = false;
175  }
176
177  // Event listeners for mouse actions
178  whiteboard.addEventListener("mousedown", (e) => {
179    if (!currentShape) return;
180    drawingShape = true;
181    startX = e.offsetX;
182    startY = e.offsetY;
183  });

```

```
185 whiteboard.addEventListener("mousemove", (e) => {
186   if (!drawingShape) return;
187   const endX = e.offsetX;
188   const endY = e.offsetY;
189   clearCanvas();
190   redrawShapes();
191   drawShape(currentShape, startX, startY, endX, endY);
192 });
193
194 whiteboard.addEventListener("mouseup", (e) => {
195   if (!drawingShape) return;
196   drawingShape = false;
197   const endX = e.offsetX;
198   const endY = e.offsetY;
199   shapes.push({ shape: currentShape, startX, startY, endX, endY }); // Store the shape data
200   redrawShapes();
201 });
202
203 // Function to draw different shapes
204 function drawShape(shape, x1, y1, x2, y2) {
205   const width = x2 - x1;
206   const height = y2 - y1;
207   ctx.beginPath();
208   switch (shape) {
209     case "square":
210       ctx.rect(x1, y1, width, width);
211       break;
212     case "circle":
213       const radius = Math.sqrt(width * width + height * height) / 2;
214       ctx.arc(x1 + width / 2, y1 + height / 2, radius, 0, Math.PI * 2);
215       break;
216     case "triangle":
217       ctx.moveTo(x1, y2);
218       ctx.lineTo(x1 + width / 2, y1);
219       ctx.lineTo(x2, y2);
220
221       ctx.closePath();
222       break;
223     case "ellipse":
224       ctx.ellipse(x1 + width / 2, y1 + height / 2, Math.abs(width) / 2, Math.abs(height) / 2, 0, 0, Math.PI * 2);
225       break;
226     case "hexagon":
227       drawPolygon(6, x1, y1, width, height);
228       break;
229     case "pentagon":
230       drawPolygon(5, x1, y1, width, height);
231       break;
232     case "star":
233       drawStar(x1 + width / 2, y1 + height / 2, Math.abs(width) / 2, Math.abs(width) / 4, 5);
234       break;
235     case "rhombus":
236       ctx.moveTo(x1 + width / 2, y1);
237       ctx.lineTo(x2, y1 + height / 2);
238       ctx.lineTo(x1 + width / 2, y2);
239       ctx.lineTo(x1, y1 + height / 2);
240       ctx.closePath();
241       break;
242     case "parallelogram":
243       ctx.moveTo(x1 + width / 4, y1);
244       ctx.lineTo(x2, y1);
245       ctx.lineTo(x2 - width / 4, y2);
246       ctx.lineTo(x1, y2);
247       ctx.closePath();
248       break;
249     case "octagon":
250       drawPolygon(8, x1, y1, width, height);
251       break;
252     case "cross":
253       drawCross(x1, y1, width, height);
254       break;
```

```
255   }
256 }
257
258 // Function to draw a polygon with n sides
259 function drawPolygon(n, x1, y1, width, height) {
260   const angle = (Math.PI * 2) / n;
261   let x = x1;
262   let y = y1;
263   const cx = x1 + width / 2;
264   const cy = y1 + height / 2;
265   const radius = Math.sqrt(width * width + height * height) / 2;
266   const startAngle = Math.atan2(y - cy, x - cx);
267   const endAngle = startAngle + angle;
268   const points = [];
269   for (let i = 0; i < n; i++) {
270     const pointX = cx + radius * Math.cos(endAngle);
271     const pointY = cy + radius * Math.sin(endAngle);
272     points.push({ x: pointX, y: pointY });
273     endAngle += angle;
274   }
275   ctx.beginPath();
276   ctx.moveTo(x1, y1);
277   for (let i = 0; i < n; i++) {
278     const point = points[i];
279     ctx.lineTo(point.x, point.y);
280   }
281   ctx.closePath();
282 }
283
284 // Function to draw a star with n points
285 function drawStar(x1, y1, width, height, points) {
286   const angle = (Math.PI * 2) / points;
287   let x = x1;
288   let y = y1;
289   const cx = x1 + width / 2;
290   const cy = y1 + height / 2;
291   const radius = Math.sqrt(width * width + height * height) / 2;
292   const startAngle = Math.atan2(y - cy, x - cx);
293   const endAngle = startAngle + angle;
294   const pointsArray = [];
295   for (let i = 0; i < points; i++) {
296     const pointX = cx + radius * Math.cos(endAngle);
297     const pointY = cy + radius * Math.sin(endAngle);
298     pointsArray.push({ x: pointX, y: pointY });
299     endAngle += angle;
300   }
301   ctx.beginPath();
302   ctx.moveTo(x1, y1);
303   for (let i = 0; i < points; i++) {
304     const point = pointsArray[i];
305     ctx.lineTo(point.x, point.y);
306   }
307   ctx.closePath();
308 }
309
310 // Function to draw a cross
311 function drawCross(x1, y1, width, height) {
312   const cx = x1 + width / 2;
313   const cy = y1 + height / 2;
314   const radius = Math.sqrt(width * width + height * height) / 2;
315   const angle = Math.PI / 2;
316   const points = [
317     { x: cx + radius * Math.cos(angle), y: cy + radius * Math.sin(angle) },
318     { x: cx - radius * Math.cos(angle), y: cy + radius * Math.sin(angle) },
319     { x: cx + radius * Math.cos(angle), y: cy - radius * Math.sin(angle) },
320     { x: cx - radius * Math.cos(angle), y: cy - radius * Math.sin(angle) }
321   ];
322   ctx.beginPath();
323   ctx.moveTo(x1, y1);
324   for (let i = 0; i < points.length; i++) {
325     const point = points[i];
326     ctx.lineTo(point.x, point.y);
327   }
328   ctx.closePath();
329 }
```

```

254     case "heart":
255         drawHeart(x1, y1, width, height);
256         break;
257     case "line":
258         ctx.moveTo(x1, y1);
259         ctx.lineTo(x2, y2);
260         break;
261     case "arrow":
262         drawArrow(x1, y1, x2, y2);
263         break;
264     }
265     ctx.stroke();
266 }
267
268 // Function to draw a polygon (for hexagons, pentagons, etc.)
269 function drawPolygon(sides, x1, y1, width, height) {
270     const centerX = x1 + width / 2;
271     const centerY = y1 + height / 2;
272     const radius = Math.min(Math.abs(width), Math.abs(height)) / 2;
273     ctx.moveTo(centerX + radius * Math.cos(0), centerY + radius * Math.sin(0));
274     for (let i = 1; i <= sides; i++) {
275         const angle = (i * 2 * Math.PI) / sides;
276         ctx.lineTo(centerX + radius * Math.cos(angle), centerY + radius * Math.sin(angle));
277     }
278     ctx.closePath();
279 }
280
281 // Function to draw a star
282 function drawStar(cx, cy, outerRadius, innerRadius, points) {
283     const step = Math.PI / points;
284     let angle = -Math.PI / 2;
285     ctx.moveTo(cx + outerRadius * Math.cos(angle), cy + outerRadius * Math.sin(angle));
286     for (let i = 0; i < points; i++) {
287         angle += step;
288         ctx.lineTo(cx + innerRadius * Math.cos(angle), cy + innerRadius * Math.sin(angle));
289
290         angle += step;
291         ctx.lineTo(cx + outerRadius * Math.cos(angle), cy + outerRadius * Math.sin(angle));
292     }
293     ctx.closePath();
294 }
295
296 // Function to draw a cross
297 function drawCross(x1, y1, width, height) {
298     const armWidth = Math.min(width, height) / 4;
299     const centerX = x1 + width / 2;
300     const centerY = y1 + height / 2;
301     // Horizontal line
302     ctx.moveTo(centerX - armWidth / 2, centerY);
303     ctx.lineTo(centerX + armWidth / 2, centerY);
304     // Vertical line
305     ctx.moveTo(centerX, centerY - armWidth / 2);
306     ctx.lineTo(centerX, centerY + armWidth / 2);
307 }
308
309 // Function to draw a heart
310 function drawHeart(x1, y1, width, height) {
311     const x2 = x1 + width;
312     const y2 = y1 + height;
313     ctx.moveTo(x1 + width / 2, y1 + height / 4);
314     ctx.bezierCurveTo(x1 + width / 2, y1, x1, y1, x1, y1 + height / 4);
315     ctx.bezierCurveTo(x1, y1 + height / 2, x1 + width / 2, y2, x1 + width / 2, y2);
316     ctx.bezierCurveTo(x1 + width / 2, y2, x2, y1 + height / 2, x2, y1 + height / 4);
317     ctx.bezierCurveTo(x2, y1, x1 + width / 2, y1, x1 + width / 2, y1 + height / 4);
318     ctx.closePath();
319 }
320
321 // Function to draw an arrow
322 function drawArrow(x1, y1, x2, y2) {
323     ctx.moveTo(x1, y1);
324     ctx.lineTo(x2, y2);

```

```

324 // Arrowhead
325 const angle = Math.atan2(y2 - y1, x2 - x1);
326 const arrowLength = 15; // Length of the arrowhead
327 const arrowAngle = Math.PI / 6; // Angle of the arrowhead
328 ctx.lineTo(x2 - arrowLength * Math.cos(angle - arrowAngle), y2 - arrowLength * Math.sin(angle - arrowAngle));
329 ctx.moveTo(x2, y2);
330 ctx.lineTo(x2 - arrowLength * Math.cos(angle + arrowAngle), y2 - arrowLength * Math.sin(angle + arrowAngle));
331 ctx.stroke();
332 }
333
334 // Function to redraw all the shapes
335 function redrawShapes() {
336 shapes.forEach((shapeData) => {
337 drawShape(shapeData.shape, shapeData.startX, shapeData.startY, shapeData.endX, shapeData.endY);
338 });
339 }
340
341 // Function to clear the canvas
342 function clearCanvas() {
343 ctx.clearRect(0, 0, whiteboard.width, whiteboard.height);
344 if (showGrid) drawGrid(); // Redraw grid if enabled
345 }

```

Java script code for Color Picker functionality:

```

403 // Function to change the brush color based on the color picker
404 function changeBrushColor(color) {
405 ctx.strokeStyle = color;
406 ctx.fillStyle = color;
407 }
408 // Event listener for color picker
409 colorPicker.addEventListener("input", () => {
410 setBrushSettings(currentBrush); // Update brush settings when color is changed
411 });
412
413 // Mouse events for drawing with brush
414 whiteboard.addEventListener("mousedown", (e) => {
415 drawing = true;
416 lastX = e.offsetX;
417 lastY = e.offsetY;
418 ctx.beginPath();
419 ctx.moveTo(lastX, lastY);
420 });
421
422 whiteboard.addEventListener("mouseup", () => {
423 drawing = false;
424 ctx.closePath();
425 });
426
427 whiteboard.addEventListener("mouseout", () => {
428 drawing = false;
429 ctx.closePath();
430 });
431
432 pencilBtn.addEventListener("click", () => {
433 currentTool = 'pencil';
434 isErasing = false;
435 ctx.strokeStyle = document.getElementById("color-picker").value; // Set to selected color
436 ctx.lineWidth = 2; // Thin line for pencil
437 });
438
439 markerBtn.addEventListener("click", () => {
440 currentTool = 'marker';
441 isErasing = false;
442 ctx.strokeStyle = document.getElementById("color-picker").value; // Set to selected color
443 ctx.lineWidth = 5; // Thick line for marker
444 });
445

```

Java Script code for Brushes functionalities:

```
347 // Brushes options
348 const crayonBtn = document.getElementById("crayonBtn");
349 const airbrushBtn = document.getElementById("airbrushBtn");
350 const watercolorBtn = document.getElementById("watercolorBtn");
351 const oilBrushBtn = document.getElementById("oilBrushBtn");
352 const calligraphyBtn = document.getElementById("calligraphyBtn");
353 const colorPicker = document.getElementById("color-picker");
354
355 // Set up brush tool properties
356 const brushSettings = {
357     crayon: { lineWidth: 15, lineCap: "round", globalAlpha: 1 },
358     airbrush: { lineWidth: 5, lineCap: "round", globalAlpha: 0.3 },
359     watercolor: { lineWidth: 10, lineCap: "round", globalAlpha: 0.5 },
360     oilBrush: { lineWidth: 20, lineCap: "round", globalAlpha: 1 },
361     calligraphy: { lineWidth: 5, lineCap: "butt", globalAlpha: 1 }
362 };
363
364 // Set brush settings based on selected tool
365 function setBrushSettings(brushType) {
366     const settings = brushSettings[brushType];
367     ctx.lineWidth = settings.lineWidth;
368     ctx.lineCap = settings.lineCap;
369     ctx.globalAlpha = settings.globalAlpha;
370     ctx.strokeStyle = colorPicker.value;
371 }
372
373 // Set default brush settings
374 let currentBrush = "crayon";
375 setBrushSettings(currentBrush);
376
377 // Brush tool event listeners
378 crayonBtn.addEventListener("click", () => {
379     currentBrush = "crayon";
380     setBrushSettings(currentBrush);
381 });
382
383 airbrushBtn.addEventListener("click", () => {
384     currentBrush = "airbrush";
385     setBrushSettings(currentBrush);
386 });
387
388 watercolorBtn.addEventListener("click", () => {
389     currentBrush = "watercolor";
390     setBrushSettings(currentBrush);
391 });
392
393 oilBrushBtn.addEventListener("click", () => {
394     currentBrush = "oilBrush";
395     setBrushSettings(currentBrush);
396 });
397
398 calligraphyBtn.addEventListener("click", () => {
399     currentBrush = "calligraphy";
400     setBrushSettings(currentBrush);
401 });
```

Java Script code for Sticky notes functionality:

```
616 //sticky notes
617 let stickyNotes = [];
618 let selectedStickyNote = null;
619 let offsetX = 0;
620 let offsetY = 0;
621
622 document.getElementById('addStickyNoteBtn').addEventListener('click', function () {
623 | addStickyNote();
624 });
625
626 // Add a sticky note to the canvas
627 function addStickyNote() {
628 | const note = {
629 | | id: Date.now(),
630 | | x: 100, // Initial position
631 | | y: 100,
632 | | width: 150,
633 | | height: 150,
634 | | text: 'New Note',
635 | | isEditing: false
636 | };
637 stickyNotes.push(note);
638 renderStickyNotes();
639 }
640
641 // Render all sticky notes on the canvas
642 function renderStickyNotes() {
643 | const canvas = document.getElementById('whiteboard');
644 | const ctx = canvas.getContext('2d');
645 | ctx.clearRect(0, 0, canvas.width, canvas.height); // Clear the canvas
646 stickyNotes.forEach(note => {
647 | | ctx.fillStyle = '#FFD700';
648 | | ctx.fillRect(note.x, note.y, note.width, note.height);
649 | | ctx.fillStyle = 'black';
650 | | ctx.font = '16px Arial';
```

```
651 | | wrapText(ctx, note.text, note.x + 10, note.y + 30, note.width - 20, note.height - 40);
652 | });
653 }
654
655 // Wrap text inside the sticky note box, ensuring it fits within the box boundaries
656 function wrapText(ctx, text, x, y, maxWidth, maxHeight) {
657 | const words = text.split(' ');
658 | let line = '';
659 | let lineHeight = 20;
660 | let currentHeight = y;
661 for (let i = 0; i < words.length; i++) {
662 | | const testLine = line + words[i] + ' ';
663 | | const testWidth = ctx.measureText(testLine).width;
664 // If the line exceeds the maxWidth, draw the current line and start a new one
665 if (testWidth > maxWidth && i > 0) {
666 | | ctx.fillText(line, x, currentHeight);
667 | | line = words[i] + ' ';
668 | | currentHeight += lineHeight;
669 } else {
670 | | line = testLine;
671 }
672 // If the current height exceeds the max height, stop drawing more text
673 if (currentHeight + lineHeight > y + maxHeight) {
674 | | ctx.fillText('...', x, currentHeight); // Indicate that the text is truncated
675 | | break;
676 }
677 }
678 // Draw the last line of text if within bounds
679 if (currentHeight + lineHeight <= y + maxHeight) {
680 | | ctx.fillText(line, x, currentHeight);
681 }
682 }
```

```
683 // Edit a sticky note
684 function editStickyNote(note) {
685 | const newText = prompt('Edit Sticky Note:', note.text);
686 if (newText != null) {
687 | | note.text = newText;
688 | | renderStickyNotes();
689 }
690 }
691 // Delete a sticky note
692 function deleteStickyNote(note) {
693 | const index = stickyNotes.indexOf(note);
694 if (index > -1) {
695 | | stickyNotes.splice(index, 1);
696 | | renderStickyNotes();
697 }
698 }
699
700 // Event listeners for interacting with sticky notes
701 document.getElementById('whiteboard').addEventListener('mousedown', function (e) {
702 | const canvas = e.target;
703 | const rect = canvas.getBoundingClientRect();
704 | const x = e.clientX - rect.left;
705 | const y = e.clientY - rect.top;
706
707 // Check if a sticky note is clicked
708 for (let i = stickyNotes.length - 1; i >= 0; i--) {
709 | | const note = stickyNotes[i];
710 if (x >= note.x && x <= note.x + note.width && y >= note.y && y <= note.y + note.height) {
711 | | | selectedStickyNote = note;
712 | | | offsetX = x - note.x;
713 | | | offsetY = y - note.y;
714
715 // If Ctrl key is pressed, delete the sticky note
716 if (e.ctrlKey) {
717 | | | deleteStickyNote(note);
718 } else {
```

```
718 | | | } else {
719 | | | // If the sticky note is clicked, edit it
720 | | | if (!note.isEditing) {
721 | | | | note.isEditing = true;
722 | | | | editStickyNote(note);
723 | | | }
724 | | }
725 | | return;
726 | }
727 }
728 });
729 // Move sticky note when dragging
730 document.getElementById('whiteboard').addEventListener('mousemove', function (e) {
731 if (selectedStickyNote) {
732 | const canvas = e.target;
733 | const rect = canvas.getBoundingClientRect();
734 | const x = e.clientX - rect.left;
735 | const y = e.clientY - rect.top;
736 | selectedStickyNote.x = x - offsetX;
737 | selectedStickyNote.y = y - offsetY;
738 | renderStickyNotes();
739 }
740 });
741
742 // Release selected sticky note when mouse button is released
743 document.getElementById('whiteboard').addEventListener('mouseup', function () {
744 | | selectedStickyNote = null;
745 });
746
747 // Initial rendering
748 renderStickyNotes();
```

JavaScript code for Background functionality:

```
777 //Background Btn
778 const backgroundColorBtn = document.getElementById("backgroundColorBtn");
779
780 // Initialize the whiteboard canvas
781 const canvas = whiteboard;
782 canvas.width = window.innerWidth;
783 canvas.height = window.innerHeight;
784 // Function to change the background color
785 function changeBackgroundColor() {
786     // Open a color picker dialog
787     const colorPicker = document.createElement("input");
788     colorPicker.type = "color";
789     colorPicker.style.display = "none";
790     document.body.appendChild(colorPicker);
791     colorPicker.click();
792
793     // Update the canvas background color when a color is selected
794     colorPicker.addEventListener("input", (event) => {
795         const selectedColor = event.target.value;
796         ctx.fillStyle = selectedColor;
797         ctx.fillRect(0, 0, canvas.width, canvas.height);
798     });
799
800     // Clean up the color picker element
801     colorPicker.addEventListener("change", () => {
802         document.body.removeChild(colorPicker);
803     });
804 }
805 backgroundColorBtn.addEventListener("click", changeBackgroundColor);
```

JavaScript code for Export functionality:

```
807 //Export functionality
808 document.getElementById("exportPdfBtn").addEventListener("click", function () {
809     const canvas = document.getElementById("whiteboard");
810     if (!canvas) {
811         alert("Whiteboard canvas not found!");
812         return;
813     }
814     // Convert canvas to an image
815     const canvasData = canvas.toDataURL("image/png");
816     // Initialize jsPDF
817     const { jsPDF } = window.jspdf;
818     const pdf = new jsPDF();
819     // Get the canvas dimensions
820     const canvasWidth = canvas.width;
821     const canvasHeight = canvas.height;
822     // Calculate PDF dimensions (A4 size: 210mm x 297mm)
823     const pageWidth = pdf.internal.pageSize.getWidth();
824     const pageHeight = (canvasHeight / canvasWidth) * pageWidth;
825     // Add the image to the PDF
826     pdf.addImage(canvasData, "PNG", 0, 0, pageWidth, pageHeight);
827     // Prompt the user to download the PDF
828     pdf.save("whiteboard.pdf");
829 });
830 // Selecting necessary elements
831 const lineWidthSlider = document.getElementById('lineWidth');
832 // Canvas and Context setup
833 canvas.width = canvas.offsetWidth;
834 canvas.height = canvas.offsetHeight;
835 // Default settings
836 let currentLineWidth = 2;
837 // Update line width based on slider value
838 lineWidthSlider.addEventListener('input', (e) => {
839     currentLineWidth = e.target.value;
840     ctx.lineWidth = currentLineWidth;
841 });
```

```
843 // Tool selection logic
844 function selectTool(toolName) {
845   currentTool = toolName;
846   switch (toolName) {
847     case 'eraser':
848       ctx.globalCompositeOperation = 'destination-out';
849       break;
850     default:
851       ctx.globalCompositeOperation = 'source-over';
852       break;
853   }
854   ctx.lineWidth = currentLineWidth;
855 }
856
857 // Tool button event listeners
858 pencilBtn.addEventListener('click', () => selectTool('pencil'));
859 eraserBtn.addEventListener('click', () => selectTool('eraser'));
860 markerBtn.addEventListener('click', () => selectTool('marker'));
861 crayonBtn.addEventListener('click', () => selectTool('crayon'));
862 airbrushBtn.addEventListener('click', () => selectTool('airbrush'));
863 oilBrushBtn.addEventListener('click', () => selectTool('oilBrush'));
864 calligraphyBtn.addEventListener('click', () => selectTool('calligraphy'));
865 watercolorBtn.addEventListener('click', () => selectTool('watercolor'));
866
867 // Drawing functionality
868 let isDrawing = false;
869
870 let isFirstMove = true;
871 // Smooth drawing
872 function drawLine(x, y) {
873   if (isFirstMove) {
874     ctx.moveTo(x, y);
875     isFirstMove = false;
876   } else {
877     ctx.lineTo(x, y);
878     ctx.stroke();
879   }
880 }
881 // Drawing event listeners
882 canvas.addEventListener('mousedown', (e) => {
883   isDrawing = true;
884   const rect = canvas.getBoundingClientRect();
885   lastX = e.clientX - rect.left;
886   lastY = e.clientY - rect.top;
887   ctx.beginPath();
888   isFirstMove = true; // Reset on mouse down
889 });
890
891 canvas.addEventListener('mousemove', (e) => {
892   if (!isDrawing) return;
893
894   const rect = canvas.getBoundingClientRect();
895   const x = e.clientX - rect.left;
896   const y = e.clientY - rect.top;
897
898   drawLine(x, y);
899 });
900
901 canvas.addEventListener('mouseup', () => {
902   isDrawing = false;
903   lastX = 0;
904   lastY = 0;
905   isFirstMove = true;
906 });
907
908 canvas.addEventListener('mouseout', () => {
909   isDrawing = false;
910   lastX = 0;
911   lastY = 0;
912   isFirstMove = true;
913 });
```

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/>