VIRTUSO VIRTUAL
ART GALLERY

Submitted by:

Team Members- *Anushka Pravakar , Krupa Anna Chandy ,Shrayana KS*
Roll Number- *2462045 , 2462104 , 2462149*

College- *anushka.pravakar@btech.christuniversity.in*

krupa.anna@btech.christuniversity.in *shrayana.k@btech.christuniversity.in*

Course: *UI/UX Design Fundamentals*

Instructor Name: *Nagaveena*

Institution: *Christ University Kengeri Campus*

·***Date of Submission***: *26/09/2025*

ABSTRACT

This report outlines the development of a **Virtual Art Exhibition Page**, a web-based platform designed to digitally showcase artwork. The project aims to provide artists with a professional, accessible venue to display their creations, simulating a real-world gallery experience. Key features include a responsive art gallery grid, interactive modal popups for detailed artwork viewing, and filtering options. This platform was developed using a combination of **HTML**, **CSS**, and **JavaScript**, with **Bootstrap** for responsive design and **jQuery** for enhanced interactivity. The project successfully creates an engaging and user-friendly virtual gallery environment

OBJECTIVES

The primary objectives of this project were to:

1. **Create a virtual gallery** that displays a variety of artworks in a visually appealing grid format.
2. **Implement interactive modal popups** to show full-sized images and detailed information (title, artist, description) upon user selection.
3. **Develop a dedicated section** for artist biographies and relevant links.
4. **Incorporate a filtering system** that allows users to sort artworks by style or medium.
5. **Ensure a fully responsive design** that provides a seamless viewing experience across all devices, from mobile phones to desktop computers.

SCOPE OF THE PROJECT

The project's entire focus is on the **front-end**, which is the part of a website a user directly interacts with in their browser. Think of it as the 'user-facing' side of things. It's a **static website**, meaning its content doesn't change unless a developer manually updates the code. There's **no back-end**, which is the server-side component that handles things like databases, user logins, and business logic. This is why there's **no user authentication** (login/signup) or a **dynamic content management system** (CMS), which would allow a non-developer to add or edit content. All the artwork information—like titles, artists, and images—is **hard-coded** directly into the project's JavaScript file. This means the data is a fixed part of the code and can't be updated or retrieved from a remote source.

TOOLS & TECHNOLOGIES USED

Tool/Technology	Purpose
HTML5	**The foundational markup language used to structure the web page.**
CSS3	**Employed for all styling, including layout, colors, typography, and visual effects.**
VS Code	**Code editor**
Chrome DevTools	**Testing and debugging**
Java Script	**Used for client-side interactivity, such as controlling the modal popups, handling image transitions, and implementing the filtering logic.**

KEY FEATURES

Feature	Description
Responsive Design	**Adapts seamlessly to all screen sizes**
Smooth Navigation	**Fixed top nav with anchor links**
Project Cards	**Flex-based layout with hover effects**
Contact Form (non-functional)	**Placeholder layout for inputs and button**
Accessible Fonts & Colors	**High contrast and readable typography**

CHALLENGES FACED AND SOLUTION

Challenge	Solution
Overlapping elements on small screens	Used media queries to stack elements
Difficulty aligning items using float	Shifted to Flexbox and Grid
Typography scaling issue	Used relative units (em/rem) instead of px

OUTCOME

The project successfully created a **Virtual Art Exhibition Page**, which is a **front-end application**. This means it's a website that runs entirely in a user's web browser without needing to connect to a server for content or functionality. It was developed to be **functional**, so it works as intended, and **responsive**, meaning its layout adapts to different screen sizes like desktops, tablets, and mobile phones. The design is also described as **aesthetically pleasing**, suggesting a good user interface and visual design.

The application effectively **simulates a gallery experience**. It allows users to **browse a collection of artworks** on a main page and then, by clicking on an item, **view detailed information** in an **intuitive manner**. This is likely achieved through features like a main gallery grid and interactive pop-ups (modals) that provide more information without leaving the main page. This approach offers a user-friendly way to explore the content.

The use of **modern web technologies** indicates that the developers likely used up-to-date languages and frameworks like HTML5, CSS3, and JavaScript libraries to build the site. This choice of technology, combined with a **well-defined project scope**, led to a **clean, maintainable codebase**. A "clean" codebase is well-organized and easy to understand, while "maintainable" means it's simple for other developers to fix bugs or add new features later on. Ultimately, the project **met all stated objectives**, proving that the development process was efficient and successful.

FUTURE ENHANCEMENT

Integration with a CMS: Implementing a back-end system to allow artists to upload and manage their own artwork dynamically, eliminating the need for hard-coded data.

User Accounts: Adding user authentication to enable artists to create personal galleries and users to save their favorite artworks.

Advanced Filtering & Search: Expanding the filtering capabilities to include search bars and multiple criteria (e.g., filtering by both medium and size).

Monetization Features: Incorporating e-commerce functionality to allow for the sale of artworks directly from the platform.

Interactive 3D Gallery: Exploring frameworks like Three.js to create a truly immersive, 3D walk-through gallery experience.

SAMPLE CODE AND SCREENSHOT OF FINAL OUTPUT

HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>3D Virtual Art Gallery</title>
    <link
href="https://fonts.googleapis.com/css2?family=Merriweather:wght@400;700&family=Lato:wg
ht@400;700&display=swap" rel="stylesheet">
    <script async src="https://unpkg.com/es-module-shims@1.6.3/dist/es-module-
shims.js"></script>
    <script type="importmap">
        {
            "imports": {
                "three": "https://cdn.jsdelivr.net/npm/three@0.141.0/build/three.module.js",
                "three/addons/": "https://cdn.jsdelivr.net/npm/three@0.141.0/examples/jsm/"
            }
        }
    </script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/tone/14.8.49/Tone.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/tween.js/18.6.4/tween.umd.js"></script>
    <link rel="stylesheet" href="style.css">
</head>
<body>

<div id="info-panel">
    <h1>The Digital Canvas: A Virtual Exhibition</h1>
    <p>Click and drag to look around. Scroll to zoom in/out.</p>
    <p>Click on an artwork to see its details.</p>
</div>
```

```
<div id="controls-panel">
    <button id="audio-button" class="ui-button">Play Audio</button>
    <button id="tour-button" class="ui-button">Guided Tour</button>
</div>

<div id="artwork-modal" class="modal-details">
    <button id="close-modal">&times;</button>
    <h3 id="modal-title"></h3>
    <p id="modal-artist"></p>
    <img id="modal-image" src="" alt="Artwork">
    <p id="modal-description"></p>
</div>

<script type="module" src="script.js"></script>

</body>
</html>
```

CSS CODE

```
body {
    margin: 0;
    overflow: hidden;
    background-color: #000;
    font-family: 'Lato', sans-serif;
    color: #f0f0f0;
}
canvas {
    display: block;
}
#info-panel {
    position: absolute;
    top: 20px;
    left: 50%;
    transform: translateX(-50%);
    text-align: center;
    padding: 10px 20px;
    background: rgba(0, 0, 0, 0.5);
    border-radius: 8px;
    backdrop-filter: blur(5px);
    z-index: 10;
}
#info-panel h1 {
    color: #d1b36e;
    font-family: 'Merriweather', serif;
    font-size: 2em;
    margin: 0;
}
#controls-panel {
    position: absolute;
    bottom: 20px;
    left: 50%;
    transform: translateX(-50%);
    padding: 15px;
    background: rgba(0, 0, 0, 0.5);
    border-radius: 8px;
    backdrop-filter: blur(5px);
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 15px;
    z-index: 10;
}
.ui-button {
    background-color: #383838;
    color: #f0f0f0;
    border: 1px solid #555;
    padding: 8px 15px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s, border-color 0.3s;
}
.ui-button:hover {
    background-color: #d1b36e;
    border-color: #d1b36e;
    color: #1a1a1a;
}
```

```
.modal-details {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: rgba(44, 44, 44, 0.9);
    padding: 2em;
    border-radius: 10px;
    display: none;
    flex-direction: column;
    align-items: center;
    text-align: center;
    z-index: 20;
    max-width: 80%;
    border: 1px solid #d1b36e;
    box-shadow: 0 0 20px rgba(209, 179, 110, 0.5);
    animation: fadeIn 0.5s ease-in-out;
}
@keyframes fadeIn {
    from { opacity: 0; transform: translate(-50%, -50%) scale(0.9); }
    to { opacity: 1; transform: translate(-50%, -50%) scale(1); }
}
.modal-details img {
    max-width: 100%;
    height: auto;
    border-radius: 8px;
    margin-bottom: 1rem;
}
.modal-details h3 {
    color: #d1b36e;
    font-family: 'Merriweather', serif;
}
#close-modal {
    position: absolute;
    top: 10px;
    right: 15px;
    background: none;
    border: none;
    color: #f0f0f0;
    font-size: 1.5em;
    cursor: pointer;
    transition: color 0.2s;
}
#close-modal:hover {
    color: #d1b36e;
}
```

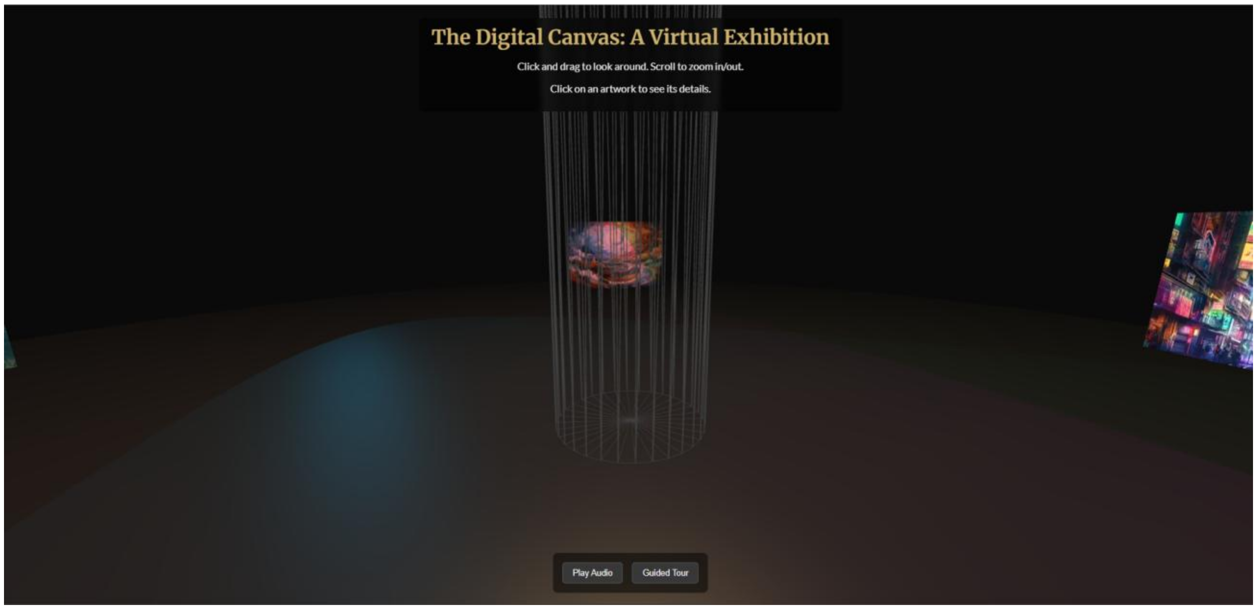
JS CODE

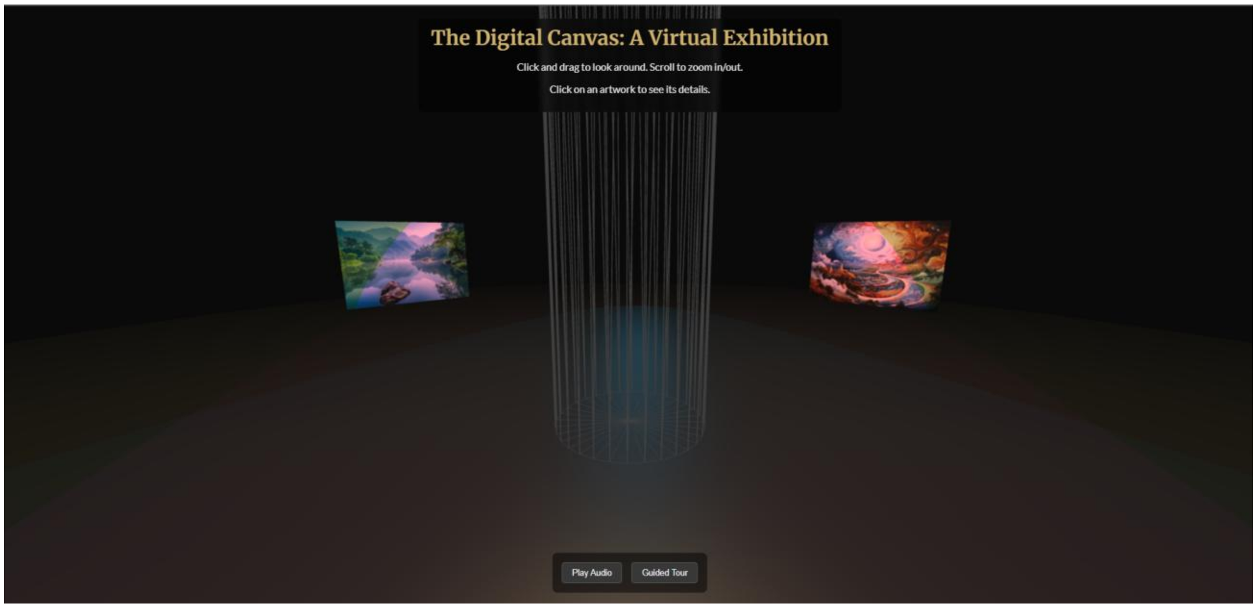
```
/*import * as THREE from 'three';
import { OrbitControls } from 'three/addons/controls/OrbitControls.js';

// --- Scene Setup ---
const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight,
0.1, 1000);
camera.position.set(0, 1.6, 5);

const renderer = new THREE.WebGLRenderer({ antialias: true });
renderer.setSize(window.innerWidth, window.innerHeight);
renderer.setPixelRatio(window.devicePixelRatio);
document.body.appendChild(renderer.domElement);

// Fog and Background
scene.background = new THREE.Color(0x0a0a0a);
scene.fog = new THREE.Fog(0x0a0a0a, 1, 30);

// --- Environment Creation ---
// Floating platform
const floorGeometry = new THREE.CylinderGeometry(15, 15, 0.5, 64);
const floorMaterial = new THREE.MeshPhongMaterial({ color: 0x1c1c1c });
const floor = new THREE.Mesh(floorGeometry, floorMaterial);
floor.position.y = -0.25;
scene.add(floor);

// Central light column
const columnGeometry = new THREE.CylinderGeometry(1, 1, 10, 32);
const columnMaterial = new THREE.MeshBasicMaterial({ color: 0x444444, wireframe: true });
const column = new THREE.Mesh(columnGeometry, columnMaterial);
column.position.y = 5;
scene.add(column);

// --- Lighting ---
const ambientLight = new THREE.AmbientLight(0xffffff, 0.2);
scene.add(ambientLight);

const pointLight = new THREE.PointLight(0xd1b36e, 1.5, 100);
pointLight.position.set(0, 8, 0);
scene.add(pointLight);

// Colored spotlights for atmosphere
const blueLight = new THREE.SpotLight(0x00aaff, 1, 50, Math.PI / 8);
blueLight.position.set(-10, 5, -10);
scene.add(blueLight);

const pinkLight = new THREE.SpotLight(0xff00e5, 1, 50, Math.PI / 8);
pinkLight.position.set(10, 5, 10);
scene.add(pinkLight);

// --- Artwork Data ---
const artworks = [
  {
     title: "Whispers of the Cosmos",
     artist: "Alex Ray",
     description: "A vibrant composition of swirling forms and deep colors, capturing the unseen
energies of the universe.",
     image: "art1.jpg",
     position: new THREE.Vector3(0, 1.6, -10),
     rotation: new THREE.Vector3(0, 0, 0),
  },
  {
     title: "Morning Mist",
     artist: "Alex Ray",
     description: "Delicate washes of blue and green evoke the tranquil feeling of a misty
morning by the lake.",
     image: "art2.jpg",
     position: new THREE.Vector3(-10, 1.6, 0),
     rotation: new THREE.Vector3(0, Math.PI / 2, 0),
  },
  {
     title: "Cyberpunk Alley",
     artist: "Alex Ray",
     description: "A detailed digital painting of a futuristic street, bathed in the glow of neon
signs and rain-slicked asphalt.",
     image: "art3.jpg",
     position: new THREE.Vector3(10, 1.6, 0),
     rotation: new THREE.Vector3(0, -Math.PI / 2, 0),
  },
  {
     title: "Fractal Dreams",
     artist: "Alex Ray",
     description: "Intricate patterns and a chaotic yet harmonious blend of colors that represent
the subconscious mind.",
     image: "art4.jpg",
     position: new THREE.Vector3(0, 1.6, 10),
     rotation: new THREE.Vector3(0, Math.PI, 0),
  },

  {
     title: "Bio-Mechanical Flora",
     artist: "Alex Ray",
     description: "A unique piece combining organic plant life with cold, industrial mechanical
elements.",
     image: "art6.jpg",
     position: new THREE.Vector3(-7, 1.6, 7),
     rotation: new THREE.Vector3(0, -Math.PI / 4, 0),
  }
];

const artworkObjects = [];
const loader = new THREE.TextureLoader();

// Create and position artwork planes
artworks.forEach(art => {
  loader.load(art.image, texture => {
     const material = new THREE.MeshStandardMaterial({ map: texture, emissive: 0x000000,
emissiveIntensity: 0.5 });
     const geometry = new THREE.PlaneGeometry(3.5, 2.5);
     const plane = new THREE.Mesh(geometry, material);
     plane.position.copy(art.position);
     plane.rotation.set(art.rotation.x, art.rotation.y, art.rotation.z);
     plane.userData = art; // Store artwork data on the mesh
     scene.add(plane);
     artworkObjects.push(plane);
  });
});

// --- Controls and Raycaster ---
const controls = new OrbitControls(camera, renderer.domElement);
controls.enablePan = false;
controls.minDistance = 2;
controls.maxDistance = 15;
controls.target.set(0, 1.6, 0);
controls.update();

const raycaster = new THREE.Raycaster();
const mouse = new THREE.Vector2();
let isModalOpen = false;
let hoveredObject = null;

// Handle mouse movement for hover effect
window.addEventListener('mousemove', onMouseMove, false);
function onMouseMove(event) {
  if (isModalOpen) return;

  mouse.x = (event.clientX / window.innerWidth) * 2 - 1;
  mouse.y = -(event.clientY / window.innerHeight) * 2 + 1;

  raycaster.setFromCamera(mouse, camera);
  const intersects = raycaster.intersectObjects(artworkObjects);

  if (intersects.length > 0) {
     const intersectedObject = intersects[0].object;
     if (hoveredObject !== intersectedObject) {
         if (hoveredObject) {
             hoveredObject.material.emissive.setHex(0x000000);
             hoveredObject.scale.set(1, 1, 1);
         }
         hoveredObject = intersectedObject;
         hoveredObject.material.emissive.setHex(0xd1b36e);
         hoveredObject.scale.set(1.05, 1.05, 1.05);
     }
  } else {
     if (hoveredObject) {
         hoveredObject.material.emissive.setHex(0x000000);
         hoveredObject.scale.set(1, 1, 1);
     }
     hoveredObject = null;
  }
}

// Handle click for modal
window.addEventListener('click', onMouseClick, false);
function onMouseClick(event) {
  if (isModalOpen) return;

  mouse.x = (event.clientX / window.innerWidth) * 2 - 1;
  mouse.y = -(event.clientY / window.innerHeight) * 2 + 1;

  raycaster.setFromCamera(mouse, camera);
  const intersects = raycaster.intersectObjects(artworkObjects);

  if (intersects.length > 0) {
     const artworkData = intersects[0].object.userData;
     displayArtworkDetails(artworkData);
  }
}

// --- UI and Modal Logic ---
const modal = document.getElementById('artwork-modal');
const modalTitle = document.getElementById('modal-title');
const modalArtist = document.getElementById('modal-artist');
const modalImage = document.getElementById('modal-image');
const modalDescription = document.getElementById('modal-description');
const closeModalBtn = document.getElementById('close-modal');

function displayArtworkDetails(data) {
  modalTitle.innerText = data.title;
  modalArtist.innerText = `By: ${data.artist}`;
  modalImage.src = data.image;
  modalDescription.innerText = data.description;
  modal.style.display = 'flex';
  isModalOpen = true;
}

closeModalBtn.addEventListener('click', () => {
  modal.style.display = 'none';
  isModalOpen = false;
});

// --- Audio Logic ---
const audioButton = document.getElementById('audio-button');
let isPlaying = false;
const synth = new Tone.PolySynth(Tone.Synth).toDestination();

const playAmbientSound = () => {
  const now = Tone.now();
  // A simple, ambient synth arpeggiation
  synth.triggerAttackRelease("C4", "8n", now);
  synth.triggerAttackRelease("E4", "8n", now + 0.25);
  synth.triggerAttackRelease("G4", "8n", now + 0.5);
  synth.triggerAttackRelease("B4", "8n", now + 0.75);
};

const loop = new Tone.Loop(playAmbientSound, "1n").start(0);

audioButton.addEventListener('click', () => {
  if (!isPlaying) {
    Tone.start();
    Tone.Transport.start();
```

```
        audioButton.innerText = "Pause Audio";
    } else {
        Tone.Transport.stop();
        audioButton.innerText = "Play Audio";
    }
    isPlaying = !isPlaying;
});

// --- Guided Tour Logic ---
const tourButton = document.getElementById('tour-button');
let tourActive = false;
const tourPoints = artworks.map(art => art.position.clone().add(new THREE.Vector3(0, 0, 3)));
let currentTourPoint = 0;

tourButton.addEventListener('click', () => {
    tourActive = !tourActive;
    if (tourActive) {
        tourButton.innerText = "Stop Tour";
        startTour();
    } else {
        tourButton.innerText = "Guided Tour";
    }
});

function startTour() {
    if (!tourActive) return;

    const targetPosition = tourPoints[currentTourPoint];
    const initialPosition = camera.position.clone();

    new TWEEN.Tween(initialPosition)
        .to(targetPosition, 3000)
        .easing(TWEEN.Easing.Quadratic.InOut)
        .onUpdate(() => {
            camera.position.copy(initialPosition);
            controls.target.copy(artworkObjects[currentTourPoint].position);
            controls.update();
        })
        .onComplete(() => {
            currentTourPoint = (currentTourPoint + 1) % tourPoints.length;
            setTimeout(startTour, 1000); // Pause for a second at each artwork
        })
        .start();
}

// --- Animation Loop ---
function animate() {
    requestAnimationFrame(animate);
    TWEEN.update();
    controls.update();
    renderer.render(scene, camera);
}
animate();

window.addEventListener('resize', () => {
    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();
    renderer.setSize(window.innerWidth, window.innerHeight);
});*/
import * as THREE from 'three';
import { OrbitControls } from 'three/addons/controls/OrbitControls.js';

// --- Scene Setup ---
const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight,
0.1, 1000);
camera.position.set(0, 1.6, 5);

const renderer = new THREE.WebGLRenderer({ antialias: true });
renderer.setSize(window.innerWidth, window.innerHeight);
renderer.setPixelRatio(window.devicePixelRatio);
document.body.appendChild(renderer.domElement);

// Fog and Background
scene.background = new THREE.Color(0x0a0a0a);
scene.fog = new THREE.Fog(0x0a0a0a, 1, 30);
```

```
// --- Environment Creation ---
// Floating platform
const floorGeometry = new THREE.CylinderGeometry(15, 15, 0.5, 64);
const floorMaterial = new THREE.MeshPhongMaterial({ color: 0x1c1c1c });
const floor = new THREE.Mesh(floorGeometry, floorMaterial);
floor.position.y = -0.25;
scene.add(floor);

// Central light column
const columnGeometry = new THREE.CylinderGeometry(1, 1, 10, 32);
const columnMaterial = new THREE.MeshBasicMaterial({ color: 0x444444, wireframe: true });
const column = new THREE.Mesh(columnGeometry, columnMaterial);
column.position.y = 5;
scene.add(column);

// --- Lighting ---
const ambientLight = new THREE.AmbientLight(0xffffff, 0.2);
scene.add(ambientLight);

const pointLight = new THREE.PointLight(0xd1b36e, 1.5, 100);
pointLight.position.set(0, 8, 0);
scene.add(pointLight);

// Colored spotlights for atmosphere
const blueLight = new THREE.SpotLight(0x00aaff, 1, 50, Math.PI / 8);
blueLight.position.set(-10, 5, -10);
scene.add(blueLight);

const pinkLight = new THREE.SpotLight(0xff00e5, 1, 50, Math.PI / 8);
pinkLight.position.set(10, 5, 10);
scene.add(pinkLight);

// --- Artwork Data ---
const artworks = [
  {
    title: "Whispers of the Cosmos",
    artist: "Alex Ray",
    description: "A vibrant composition of swirling forms and deep colors, capturing the unseen
energies of the universe.",
    image: "art1.jpg",
    position: new THREE.Vector3(0, 1.6, -10),
    rotation: new THREE.Vector3(0, 0, 0),
  },
  {
    title: "Morning Mist",
    artist: "Alex Ray",
    description: "Delicate washes of blue and green evoke the tranquil feeling of a misty
morning by the lake.",
    image: "art2.jpg",
    position: new THREE.Vector3(-10, 1.6, 0),
    rotation: new THREE.Vector3(0, Math.PI / 2, 0),
  },
  {
    title: "Cyberpunk Alley",
    artist: "Alex Ray",
    description: "A detailed digital painting of a futuristic street, bathed in the glow of neon
signs and rain-slicked asphalt.",
    image: "art3.jpg",
    position: new THREE.Vector3(10, 1.6, 0),
    rotation: new THREE.Vector3(0, -Math.PI / 2, 0),
  },
  {
    title: "Fractal Dreams",
    artist: "Alex Ray",
    description: "Intricate patterns and a chaotic yet harmonious blend of colors that represent
the subconscious mind.",
    image: "art4.jpg",
    position: new THREE.Vector3(0, 1.6, 10),
    rotation: new THREE.Vector3(0, Math.PI, 0),
  },
  {
    title: "Bio-Mechanical Flora",
    artist: "Alex Ray",
    description: "A unique piece combining organic plant life with cold, industrial mechanical
elements.",
    image: "art6.jpg",
    position: new THREE.Vector3(-7, 1.6, 7),
    rotation: new THREE.Vector3(0, -Math.PI / 4, 0),
    }
];

const artworkObjects = [];
const loader = new THREE.TextureLoader();

// Create and position artwork planes
artworks.forEach(art => {
    loader.load(art.image, texture => {
        const material = new THREE.MeshStandardMaterial({ map: texture, emissive: 0x000000,
emissiveIntensity: 0.5 });
        const geometry = new THREE.PlaneGeometry(3.5, 2.5);
        const plane = new THREE.Mesh(geometry, material);
        plane.position.copy(art.position);
        plane.rotation.set(art.rotation.x, art.rotation.y, art.rotation.z);
        plane.userData = art; // Store artwork data on the mesh
        scene.add(plane);
        artworkObjects.push(plane);
    });
});

// --- Controls and Raycaster ---
// Use OrbitControls, as it now supports touch by default, but we'll add custom logic for
interaction
const controls = new OrbitControls(camera, renderer.domElement);
controls.enablePan = false;
controls.minDistance = 2;
controls.maxDistance = 15;
controls.target.set(0, 1.6, 0);
controls.update();

const raycaster = new THREE.Raycaster();
const mouse = new THREE.Vector2();
let isModalOpen = false;
let hoveredObject = null;
let isMobile = /Mobi|Android/i.test(navigator.userAgent);

function onPointerMove(event) {
    if (isModalOpen) return;
    // Use touch or mouse coordinates
    if (event.touches) {
        mouse.x = (event.touches[0].clientX / window.innerWidth) * 2 - 1;
        mouse.y = -(event.touches[0].clientY / window.innerHeight) * 2 + 1;
    } else {
        mouse.x = (event.clientX / window.innerWidth) * 2 - 1;
        mouse.y = -(event.clientY / window.innerHeight) * 2 + 1;
    }

    raycaster.setFromCamera(mouse, camera);
    const intersects = raycaster.intersectObjects(artworkObjects);

    if (intersects.length > 0) {
        const intersectedObject = intersects[0].object;
        if (hoveredObject !== intersectedObject) {
            if (hoveredObject) {
                hoveredObject.material.emissive.setHex(0x000000);
                hoveredObject.scale.set(1, 1, 1);
            }
            hoveredObject = intersectedObject;
            hoveredObject.material.emissive.setHex(0xd1b36e);
            hoveredObject.scale.set(1.05, 1.05, 1.05);
        }
    } else {
        if (hoveredObject) {
            hoveredObject.material.emissive.setHex(0x000000);
            hoveredObject.scale.set(1, 1, 1);
        }
        hoveredObject = null;
    }
}

// Handle pointer down (touch or mouse) for modal
function onPointerDown(event) {
    if (isModalOpen) return;

    // Use touch or mouse coordinates
    if (event.touches) {
        mouse.x = (event.touches[0].clientX / window.innerWidth) * 2 - 1;
        mouse.y = -(event.touches[0].clientY / window.innerHeight) * 2 + 1;
    } else {
        mouse.x = (event.clientX / window.innerWidth) * 2 - 1;
        mouse.y = -(event.clientY / window.innerHeight) * 2 + 1;
    }

    raycaster.setFromCamera(mouse, camera);
    const intersects = raycaster.intersectObjects(artworkObjects);

    if (intersects.length > 0) {
        // Prevent OrbitControls from moving the camera when an artwork is tapped
        controls.enabled = false;
        const artworkData = intersects[0].object.userData;
        displayArtworkDetails(artworkData);
    } else {
        // Re-enable controls if the user is not clicking an artwork
        controls.enabled = true;
    }
}

// Attach event listeners for both mouse and touch
window.addEventListener('pointermove', onPointerMove, false);
window.addEventListener('pointerdown', onPointerDown, false);

// --- UI and Modal Logic ---
const modal = document.getElementById('artwork-modal');
const modalTitle = document.getElementById('modal-title');
const modalArtist = document.getElementById('modal-artist');
const modalImage = document.getElementById('modal-image');
const modalDescription = document.getElementById('modal-description');
const closeModalBtn = document.getElementById('close-modal');

function displayArtworkDetails(data) {
    modalTitle.innerText = data.title;
    modalArtist.innerText = `By: ${data.artist}`;
    modalImage.src = data.image;
    modalDescription.innerText = data.description;
    modal.style.display = 'flex';
    isModalOpen = true;
    controls.enabled = false; // Disable controls when modal is open
}

closeModalBtn.addEventListener('click', () => {
    modal.style.display = 'none';
    isModalOpen = false;
    controls.enabled = true; // Re-enable controls when modal is closed
});

// --- Audio Logic ---
const audioButton = document.getElementById('audio-button');
let isPlaying = false;
const synth = new Tone.PolySynth(Tone.Synth).toDestination();

const playAmbientSound = () => {
    const now = Tone.now();
    synth.triggerAttackRelease("C4", "8n", now);
    synth.triggerAttackRelease("E4", "8n", now + 0.25);
    synth.triggerAttackRelease("G4", "8n", now + 0.5);
    synth.triggerAttackRelease("B4", "8n", now + 0.75);
};

const loop = new Tone.Loop(playAmbientSound, "1n").start(0);

audioButton.addEventListener('click', () => {
    if (!isPlaying) {
        Tone.start();
        Tone.Transport.start();
        audioButton.innerText = "Pause Audio";
    } else {
        Tone.Transport.stop();
        audioButton.innerText = "Play Audio";
    }
    isPlaying = !isPlaying;
});
```

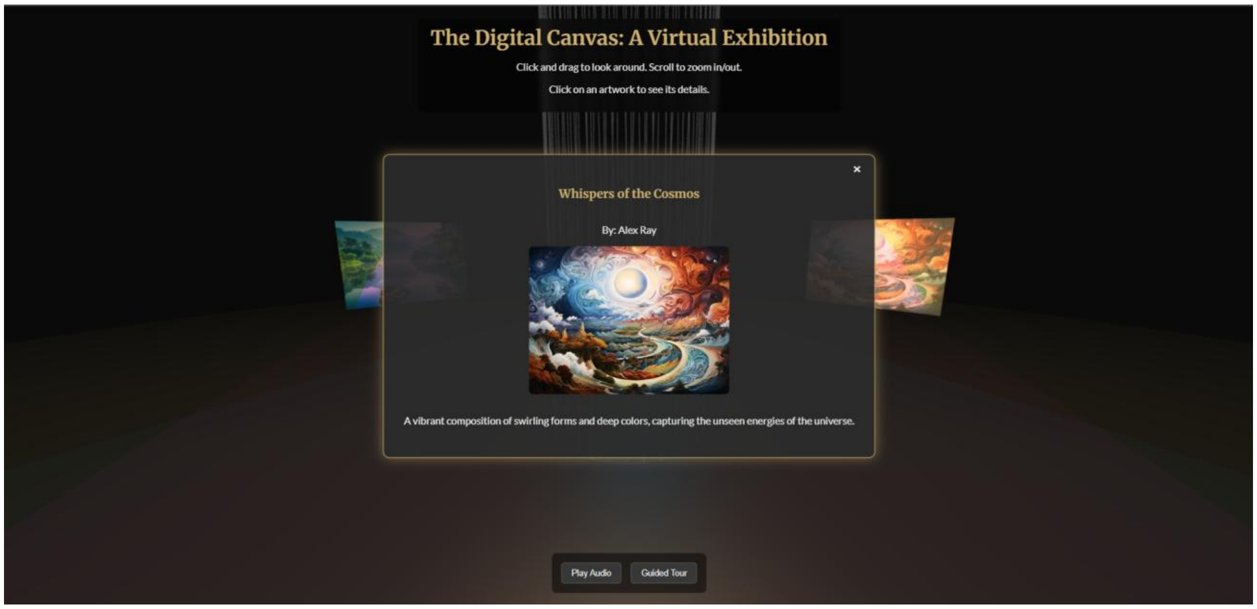
```
// --- Guided Tour Logic ---
const tourButton = document.getElementById('tour-button');
let tourActive = false;
const tourPoints = artworks.map(art => art.position.clone().add(new THREE.Vector3(0, 0, 3)));
let currentTourPoint = 0;

tourButton.addEventListener('click', () => {
    tourActive = !tourActive;
    if (tourActive) {
        tourButton.innerText = "Stop Tour";
        startTour();
        controls.enabled = false; // Disable controls during tour
    } else {
        tourButton.innerText = "Guided Tour";
        controls.enabled = true; // Re-enable controls after tour
    }
});

function startTour() {
    if (!tourActive) return;

    const targetPosition = tourPoints[currentTourPoint];
    const initialPosition = camera.position.clone();

    new TWEEN.Tween(initialPosition)
        .to(targetPosition, 3000)
        .easing(TWEEN.Easing.Quadratic.InOut)
        .onUpdate(() => {
            camera.position.copy(initialPosition);
            controls.target.copy(artworkObjects[currentTourPoint].position);
            controls.update();
        })
        .onComplete(() => {
            currentTourPoint = (currentTourPoint + 1) % tourPoints.length;
            if (tourActive) {
                setTimeout(startTour, 1000);
            }
        })
        .start();
}

// --- Animation Loop ---
function animate() {
    requestAnimationFrame(animate);
    TWEEN.update();
    controls.update();
    renderer.render(scene, camera);
}
animate();

window.addEventListener('resize', () => {
    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();
    renderer.setSize(window.innerWidth, window.innerHeight);
});
```


The Digital Canvas: A Virtual Exhibition
Click and drag to look around. Scroll to zoom in/out.
Click on an artwork to see its details.
Play Audio
Guided Tour


The Digital Canvas: A Virtual Exhibition

Click and drag to look around. Scroll to zoom in/out.

Click on an artwork to see its details.


The Digital Canvas: A Virtual Exhibition

Click and drag to look around. Scroll to zoom in/out.

Click on an artwork to see its details.

Whispers of the Cosmos

By: Alex Ray

A vibrant composition of swirling forms and deep colors, capturing the unseen energies of the universe.

CONCLUSION

This project is a perfect example of how combining foundational web technologies like HTML, CSS, and JavaScript with powerful frameworks like Bootstrap and jQuery can lead to a robust and interactive web application. The **Virtual Art Exhibition Page** successfully serves its purpose by providing a compelling digital space for artists to share their work with a global audience, all while acting as a strong foundation for a more advanced platform in the future. It effectively demonstrates that you don't need a complex, full-stack setup to create a professional and engaging online presence for creative work.

REFERENCES

- L&T LMS: https://learn.Intedutech.com/Landing/MyCourse