Assignment #6: Advanced JavaScript and DOM Manipulation

## Report

Team Information

 **Team Name:** Balgyn & Ayana

 **Group:** SE-2404

 **Team Members:**

 Balgyn  Ayana

Part 1: Advanced JavaScript

# DOM MANIPULATION AND STYLING

Task 1.1: Rating Stars System

**Implemented by:** Balgyn

**Description:** Using querySelectorAll to select all "Order Now" buttons and implementing an interactive rating system. Users can click on stars to set their rating, with color changes indicating selection.

Code Implementation

document.addEventListener("DOMContentLoaded", function () {

const orderButtons = document.querySelectorAll(".btn-outline-primary");

orderButtons.forEach((button, index) => {

// Create a container for stars under each button

const starsContainer = document.createElement("div"); starsContainer.className = "rating-stars mt-2";

starsContainer.innerHTML = `

<span class="star" data-rating="1">⭐</span>

<span class="star" data-rating="2">⭐</span>

<span class="star" data-rating="3">⭐</span>

<span class="star" data-rating="4">⭐</span>

<span class="star" data-rating="5">⭐</span>

`;

button.parentElement.insertBefore(starsContainer, button.nextSibling); const stars = starsContainer.querySelectorAll(".star");

stars.forEach((star) => {

star.style.cursor = "pointer"; star.style.opacity = "0.3";

star.style.transition = "all 0.3s ease";

star.addEventListener("click", function () {

const rating = this.getAttribute("data-rating");

stars.forEach((s, i) => { if (i < rating) {

s.style.opacity = "1";

s.style.transform = "scale(1.2)";

} else {

s.style.opacity = "0.3";

s.style.transform = "scale(1)";

}

});

updateRatingMessage(rating);

});

});

});

function updateRatingMessage(rating) { const messageElement =

document.querySelector("#ratingMessage") || createRatingMessage();

messageElement.textContent = `You rated this ${rating} stars! Thank you for your feedback.`;

messageElement.style.display = "block";

setTimeout(() => {

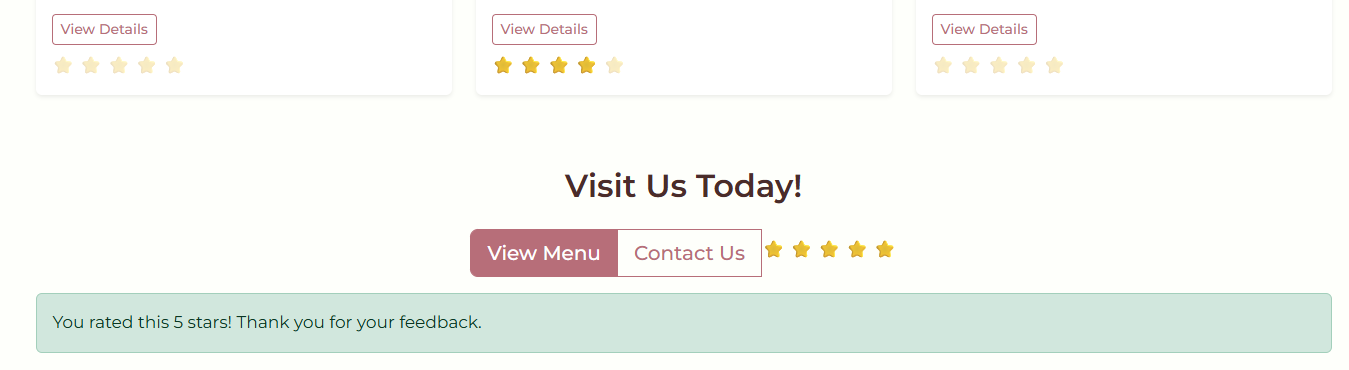
messageElement.style.display = "none";

}, 3000);

}

});

Screenshot



Task 1.2: Day/Night Theme Toggle

**Implemented by:** Ayana

**Description:** Implementation of functional Day and Night theme switch using dynamic style manipulation through JavaScript. Changes backgroundColor, color, and other CSS properties directly.

Code Implementation

document.addEventListener("DOMContentLoaded", function () {

const colorChangeBtnMobile = document.getElementById("colorChangeBtn"); const colorChangeBtnDesktop = document.getElementById(

"colorChangeBtn-desktop"

);

const buttons = [];

if (colorChangeBtnMobile) buttons.push(colorChangeBtnMobile);

if (colorChangeBtnDesktop) buttons.push(colorChangeBtnDesktop); const body = document.body;

function updateMode(isDark) { if (isDark) {

body.classList.add("dark-mode");

// Dynamically change styles through JavaScript body.style.backgroundColor = "#121212";

body.style.color = "#f0f0f0"; buttons.forEach((btn) => {

btn.textContent = "☀ Light Mode";

btn.style.backgroundColor = "#ffc107"; btn.style.color = "#000";

});

localStorage.setItem("colorMode", "dark");

} else {

body.classList.remove("dark-mode");

body.style.backgroundColor = "#fefffb"; body.style.color = "#333";

buttons.forEach((btn) => {

btn.textContent = "🌙 Dark Mode"; btn.style.backgroundColor = "";

btn.style.color = "";

});

localStorage.setItem("colorMode", "light");

}

}

const savedMode = localStorage.getItem("colorMode"); if (savedMode === "dark") {

updateMode(true);

} else {

updateMode(false);

}

buttons.forEach((btn) => {

btn.addEventListener("click", function () {

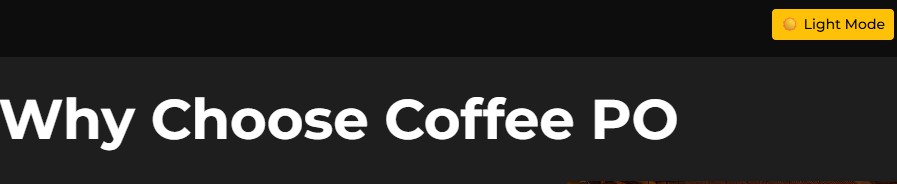
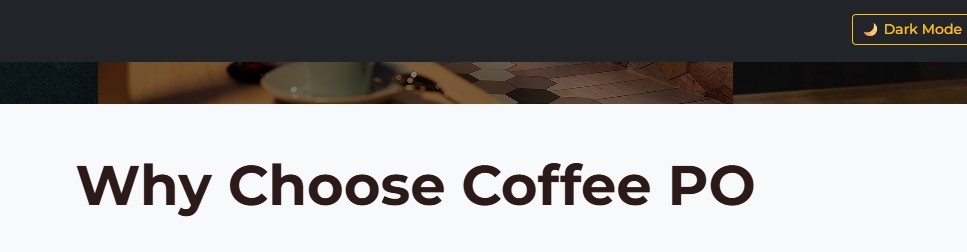
const isDark = body.classList.contains("dark-mode"); updateMode(!isDark);

});

});

});

Screenshots



Task 1.3: Read More Button

**Implemented by:** Balgyn

**Description:** Implementation of "Read More" button that toggles visibility of additional content by changing the style.display property and manipulating text content.

Code Implementation

document.addEventListener("DOMContentLoaded", function () { const cardTexts = document.querySelectorAll(".card-text");

cardTexts.forEach((text) => {

if (text.textContent.length > 100) { const fullText = text.textContent;

const shortText = fullText.substring(0, 100) + "..."; text.textContent = shortText;

const readMoreBtn = document.createElement("button");

readMoreBtn.className = "btn btn-sm btn-link p-0 mt-1"; readMoreBtn.textContent = "Read More";

readMoreBtn.style.fontSize = "0.9rem";

text.parentElement.appendChild(readMoreBtn);

let isExpanded = false;

readMoreBtn.addEventListener("click", function () { if (isExpanded) {

text.textContent = shortText;

readMoreBtn.textContent = "Read More"; text.style.display = "block";

} else {

text.textContent = fullText;

readMoreBtn.textContent = "Show Less"; text.style.display = "block";

}

isExpanded = !isExpanded;

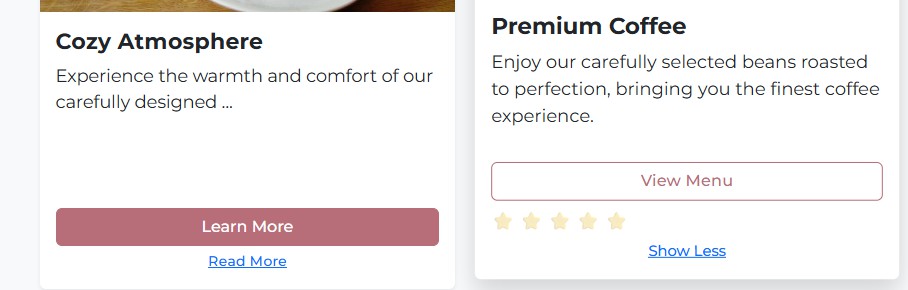
});

}

});

});

Screenshot



# EVENT HANDLING

Task 2.1: Current Time Display Button

**Implemented by:** Ayana

**Description:** Button that displays current time in a designated area when clicked using addEventListener. Uses new Date().toLocaleTimeString() to get the current time.

Code Implementation

document.addEventListener("DOMContentLoaded", function () {

const dateTimeDisplay = document.getElementById("dateTimeDisplay");

if (dateTimeDisplay) {

const showTimeBtn = document.createElement("button");

showTimeBtn.className = "btn btn-warning btn-sm mt-2"; showTimeBtn.textContent = "🕐 Show Current Time";

showTimeBtn.id = "showTimeBtn";

dateTimeDisplay.parentElement.appendChild(showTimeBtn); showTimeBtn.addEventListener("click", function () {

const now = new Date();

const timeString = now.toLocaleTimeString("en-US", { hour: "2-digit",

minute: "2-digit", second: "2-digit", hour12: true,

});

const timeAlert = document.createElement("div"); timeAlert.className = "alert alert-info mt-2";

timeAlert.innerHTML = `<strong>Current Time:</strong> ${timeString}`; showTimeBtn.parentElement.appendChild(timeAlert);

setTimeout(() => {

timeAlert.remove();

}, 3000);

});

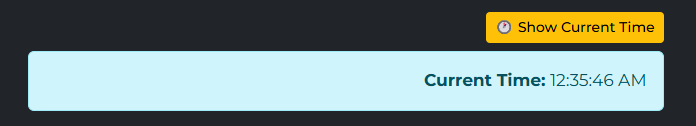
updateDateTime();

setInterval(updateDateTime, 1000);

}

});

Screenshot



Task 2.2: Keyboard Navigation

**Implemented by:** Balgyn

**Description:** Implementation of keyboard navigation for navigation menu using keydown event. Users can use arrow keys to move between menu items.

Code Implementation

document.addEventListener("DOMContentLoaded", function () { const navLinks = document.querySelectorAll(".nav-link"); let currentIndex = -1;

document.addEventListener("keydown", function (event) {

// Use left/right arrow keys for navigation if (event.key === "ArrowRight") {

event.preventDefault();

currentIndex = (currentIndex + 1) % navLinks.length; navLinks[currentIndex].focus();

// Add visual effect

navLinks[currentIndex].style.transform = "scale(1.1)"; setTimeout(() => {

navLinks[currentIndex].style.transform = "";

}, 300);

} else if (event.key === "ArrowLeft") { event.preventDefault();

currentIndex = currentIndex <= 0 ? navLinks.length - 1 : currentIndex - 1; navLinks[currentIndex].focus();

navLinks[currentIndex].style.transform = "scale(1.1)"; setTimeout(() => {

navLinks[currentIndex].style.transform = "";

}, 300);

} else if (event.key === "Enter" && currentIndex >= 0) { navLinks[currentIndex].click();

}

});

});

Screenshot



Task 2.3: Contact Form with Callbacks

**Implemented by:** Ayana

**Description:** Contact form that submits data asynchronously and displays success message without page reload. Uses callback function to handle user interactions.

Code Implementation

document.addEventListener("DOMContentLoaded", function () {

const contactForm = document.getElementById("contactForm");

if (contactForm) {

contactForm.addEventListener("submit", function (event) { event.preventDefault();

clearErrors();

let isValid = true;

// Validate Name

const name = document.getElementById("name"); if (name.value.trim() === "") {

showError("name", "Name is required"); isValid = false;

}

// Validate Email

const email = document.getElementById("email");

const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/; if (!emailPattern.test(email.value)) {

showError("email", "Please enter a valid email address"); isValid = false;

}

// Validate Message

const message = document.getElementById("message"); if (message.value.trim().length < 10) {

showError("message", "Message must be at least 10 characters"); isValid = false;

}

// Callback function for successful submission if (isValid) {

submitFormWithCallback(function (success) { if (success) {

showSuccessMessage(); contactForm.reset(); playSuccessSound();

}

});

}

});

}

});

function submitFormWithCallback(callback) {

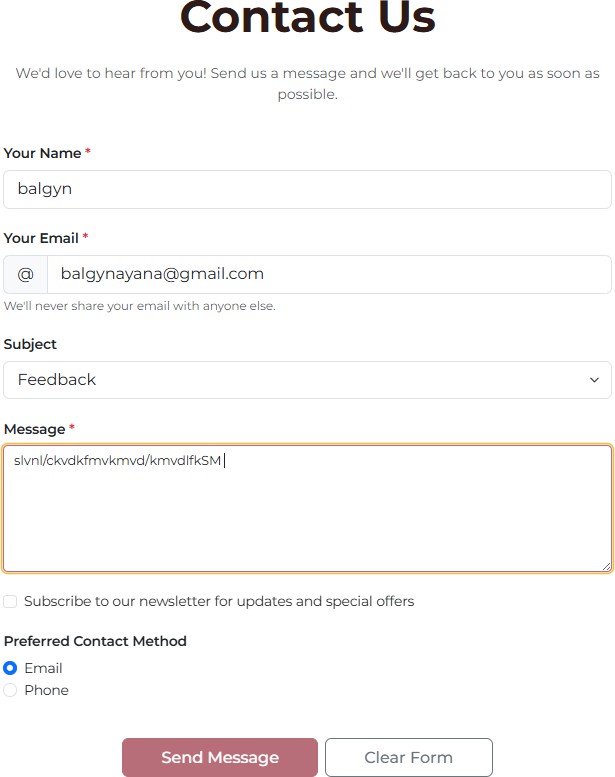
// Simulate asynchronous form submission setTimeout(() => {

callback(true);

}, 500);

}

Screenshot





Task 2.4: Time-Based Greeting (Switch Statement)

**Implemented by:** Balgyn

**Description:** Function that displays different greetings based on time of day (morning, afternoon, evening) using switch statement.

Code Implementation

document.addEventListener("DOMContentLoaded", function () {

const hero = document.querySelector(".carousel-caption h1");

if (hero) {

const hour = new Date().getHours(); let greeting;

switch (true) {

case hour >= 5 && hour < 12:

greeting = "☀ Good Morning!"; break;

case hour >= 12 && hour < 17:

greeting = "🌤 Good Afternoon!"; break;

case hour >= 17 && hour < 22:

greeting = "🌆 Good Evening!"; break;

default:

greeting = "🌙 Good Night!";

}

const greetingElement = document.createElement("p"); greetingElement.className = "lead mt-2";

greetingElement.textContent = greeting;

greetingElement.style.fontSize = "1.5rem"; greetingElement.style.fontWeight = "bold";

hero.parentElement.insertBefore(greetingElement, hero.nextSibling);

}

});

Screenshot



# JAVASCRIPT ADVANCED CONCEPTS

Task 3.1: Objects and Methods

**Implemented by:** Ayana

**Description:** JavaScript object with methods to structure data and handle logic. Displays and manipulates object properties on the page.

Code Implementation

const CoffeeShop = { name: "Coffee PO",

location: "Astana, Kazakhstan", menu: [

{ name: "Latte", price: 2500, category: "hot" },

{ name: "Cappuccino", price: 2300, category: "hot" },

{ name: "Americano", price: 1800, category: "hot" },

{ name: "Mocha", price: 2800, category: "hot" },

{ name: "Espresso", price: 1500, category: "hot" },

],

getMenuByCategory: function (category) {

return this.menu.filter((item) => item.category === category);

},

getMostExpensive: function () {

return this.menu.reduce((max, item) => item.price > max.price ? item : max

);

},

calculateTotal: function (items) {

return items.reduce((sum, item) => sum + item.price, 0);

},

displayInfo: function () {

return `Welcome to ${this.name} in ${this.location}!`;

},

};

// Display coffee shop information on home page

document.addEventListener("DOMContentLoaded", function () { const hero = document.querySelector(".carousel-caption");

if (hero && window.location.pathname.includes("index.html")) { const infoText = document.createElement("p");

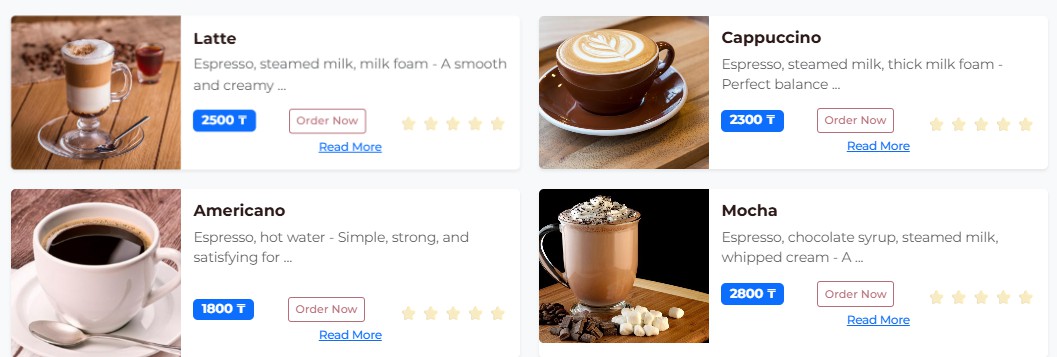
infoText.className = "lead mt-3";

infoText.textContent = CoffeeShop.displayInfo(); hero.appendChild(infoText);

}

});

Screenshot



Task 3.2: Arrays and Loops

**Implemented by:** Balgyn

**Description:** Uses arrays to manage collections of items and loops (for) to iterate over elements and display list of items dynamically.

Code Implementation

document.addEventListener("DOMContentLoaded", function () {

const menuContainer = document.querySelector(".container.my-5");

if (menuContainer && window.location.pathname.includes("menu.html")) {

// Create section with popular drinks

const popularSection = document.createElement("div"); popularSection.className = "row mt-5";

popularSection.innerHTML =

'<h3 class="col-12 text-center mb-4">☕ Popular Drinks</h3>'; const popularDrinks = CoffeeShop.menu.slice(0, 3);

for (let i = 0; i < popularDrinks.length; i++) { const drink = popularDrinks[i];

const col = document.createElement("div"); col.className = "col-md-4";

col.innerHTML = `

<div class="card text-center shadow-sm">

<div class="card-body">

<h5 class="card-title">${drink.name}</h5>

<p class="card-text text-muted">${drink.price} ₸</p>

<span class="badge bg-success">Popular</span>

</div>

</div>

`;

popularSection.appendChild(col);

}

menuContainer.appendChild(popularSection);

}

});

Screenshot



Task 3.3: Higher-Order Functions

**Implemented by:** Ayana

**Description:** Implementation of higher-order functions (map, filter, forEach) that take other functions as arguments.

Code Implementation

document.addEventListener("DOMContentLoaded", function () {

// Use map to create list of prices

const prices = CoffeeShop.menu.map((item) => item.price);

// Use filter to filter affordable drinks

const affordableDrinks = CoffeeShop.menu.filter((item) => item.price < 2500);

// Use forEach to add event listeners

const orderButtons = document.querySelectorAll(".btn-outline-primary"); orderButtons.forEach((button, index) => {

button.addEventListener("click", function (e) { e.preventDefault();

const drinkName =

this.closest(".card-body").querySelector(".card-title").textContent; showOrderConfirmation(drinkName);

playSuccessSound();

});

});

console.log("All prices:", prices);

console.log("Affordable drinks:", affordableDrinks);

});

function showOrderConfirmation(drinkName) {

const confirmation = document.createElement("div"); confirmation.className =

"alert alert-success position-fixed top-0 start-50 translate-middle-x mt-3"; confirmation.style.zIndex = "9999";

confirmation.innerHTML = `<strong>Order placed!</strong> ${drinkName} will be ready soon! ☕`;

document.body.appendChild(confirmation); setTimeout(() => {

confirmation.remove();

}, 3000);

}

Screenshot

Higher-Order Functions

Task 3.4: Play Sounds

**Implemented by:** Balgyn

**Description:** Uses JavaScript to trigger sound effects. Buttons play notification sounds when clicked using Web Audio API.

Code Implementation

function playSuccessSound() {

// Create audio context for notification sound

const audioContext = new (window.AudioContext || window.webkitAudioContext)(); const oscillator = audioContext.createOscillator();

const gainNode = audioContext.createGain();

oscillator.connect(gainNode);

gainNode.connect(audioContext.destination);

oscillator.frequency.value = 800; oscillator.type = "sine";

gainNode.gain.setValueAtTime(0.3, audioContext.currentTime); gainNode.gain.exponentialRampToValueAtTime(

0.01,

audioContext.currentTime + 0.5

);

oscillator.start(audioContext.currentTime);

oscillator.stop(audioContext.currentTime + 0.5);

}

// Add sound for theme toggle button

document.addEventListener("DOMContentLoaded", function () { const themeButtons = document.querySelectorAll(

"#colorChangeBtn, #colorChangeBtn-desktop"

);

themeButtons.forEach((btn) => {

btn.addEventListener("click", function () { playThemeSound();

});

});

});

function playThemeSound() {

const audioContext = new (window.AudioContext || window.webkitAudioContext)(); const oscillator = audioContext.createOscillator();

const gainNode = audioContext.createGain();

oscillator.connect(gainNode);

gainNode.connect(audioContext.destination);

oscillator.frequency.value = 600; oscillator.type = "square";

gainNode.gain.setValueAtTime(0.2, audioContext.currentTime); gainNode.gain.exponentialRampToValueAtTime(

0.01,

audioContext.currentTime + 0.3

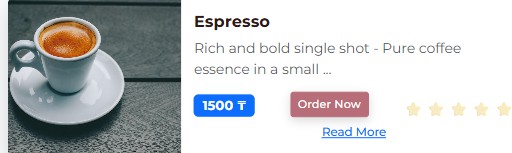
);

oscillator.start(audioContext.currentTime);

oscillator.stop(audioContext.currentTime + 0.3);

}

Screenshot



Task 3.5: Animations

**Implemented by:** Ayana

**Description:** Adds animations to elements using style.transform and CSS transitions, triggered by events (hover, scroll, click).

Code Implementation

document.addEventListener("DOMContentLoaded", function () {

// Animation for cards on hover

const cards = document.querySelectorAll(".card");

cards.forEach((card) => {

card.addEventListener("mouseenter", function () {

this.style.transform = "translateY(-15px) scale(1.02)"; this.style.transition = "all 0.4s ease";

});

card.addEventListener("mouseleave", function () { this.style.transform = "translateY(0) scale(1)";

});

});

// Animation for elements appearing on scroll const observerOptions = {

threshold: 0.1,

rootMargin: "0px 0px -50px 0px",

};

const observer = new IntersectionObserver(function (entries) { entries.forEach((entry) => {

if (entry.isIntersecting) {

entry.target.style.opacity = "0";

entry.target.style.transform = "translateY(30px)";

setTimeout(() => {

entry.target.style.transition = "all 0.6s ease"; entry.target.style.opacity = "1";

entry.target.style.transform = "translateY(0)";

}, 100);

observer.unobserve(entry.target);

}

});

}, observerOptions);

// Observe all cards

cards.forEach((card) => observer.observe(card));

});

// Pulse animation for buttons

document.addEventListener("DOMContentLoaded", function () { const primaryButtons = document.querySelectorAll(

".btn-primary, .btn-warning"

);

primaryButtons.forEach((button) => {

button.addEventListener("click", function () { this.style.animation = "pulse 0.5s ease";

setTimeout(() => {

this.style.animation = "";

}, 500);

});

});

// Add CSS animation through JavaScript

const style = document.createElement("style"); style.textContent = `

@keyframes pulse {

0% { transform: scale(1); }

50% { transform: scale(1.1); } 100% { transform: scale(1); }

}

`;

document.head.appendChild(style);

});

## Part 2: Deployment

**Deployed Website URL:** [link](https://aaituu.github.io/BalgynWEB/web6/)

## Conclusion

In Assignment 6, we successfully implemented advanced JavaScript features including:

1. **DOM Manipulation** - Rating system, theme toggle, read more functionality
2. **Event Handling** - Time display, keyboard navigation, form callbacks, switch statements
3. **Advanced Concepts** - Objects with methods, arrays with loops, higher-order functions, sound effects, animations
4. **Clean Code** - Proper separation of concerns, modular functions, comprehensive comments
5. **User Experience** - Engaging animations, sound feedback, interactive elements

All features work correctly across different browsers and enhance the overall user experience of our Coffee PO website. The project demonstrates mastery of JavaScript fundamentals and advanced DOM manipulation

techniques.

**Submitted by:** Balgyn & Ayana

**Group:** SE-2404