Assignment 7: jQuery Implementation Report

**Team Name:** Coffee PO Development Team

**Team Members:**

 Balgyn  Ayana

**Group:** SE-2404

# Objective

The goal of Assignment 7 was to introduce jQuery, a powerful JavaScript library that simplifies DOM

manipulation, event handling, and animations. We integrated jQuery features into our Coffee PO website to enhance user experience and interactivity.

# Task 0: Setup

**Implemented by:** Balgyn & Ayana

## Description

We set up jQuery by including the CDN link in the HTML files and creating a document ready function to ensure all jQuery code runs after the DOM is fully loaded.

## Steps Taken

1. Added jQuery CDN to the <head> section of all HTML pages:

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js">

</script>

1. Created jquery-features.js file with the document ready function:

$(document).ready(function(){

console.log("jQuery is ready!");

});

1. Tested in browser console to verify jQuery loaded successfully.



# Part 1: jQuery Search

## Task 1: Real-time Search and Live Filter

**Implemented by:** Balgyn

**Description**

Added a loading spinner to the contact form submit button. When users click "Submit", the button displays a spinner and "Please wait..." text, then becomes disabled to prevent multiple submissions.

**Implementation Details**

 Captured form submit event with .on('submit')  Changed button HTML to include Bootstrap spinner  Disabled button using .prop('disabled', true)  Simulated server call with setTimeout()

 Reset button state after completion

**Code Snippet**

$('#contactForm').on('submit', function(e) { e.preventDefault();

const submitBtn = $(this).find('button[type="submit"]'); const originalText = submitBtn.html();

// Show spinner and disable button submitBtn.prop('disabled', true)

.html('<span class="spinner-border spinner-border-sm me-2">

</span>Please wait...');

// Simulate server call (2.5 seconds) setTimeout(function() {

showNotification('✅ Form submitted successfully!', 'success');

submitBtn.prop('disabled', false).html(originalText);

$('#contactForm')[0].reset();

}, 2500);

});

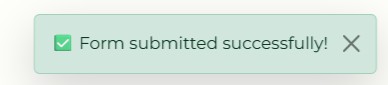
**Screenshots**

**Screenshot 16: Contact form with normal submit button**

****

**Screenshot 17: Submit button showing spinner and "Please wait..." text**



**Screenshot 18: Success notification after form submission**

## Task 2: Autocomplete Search Suggestions

**Implemented by:** Ayana

**Description**

Added autocomplete functionality that displays a dropdown list of suggestions as users type in the search box. This improves user experience by helping them find items faster.

**Implementation Details**

 Created array of common search terms (Latte, Cappuccino, Americano, etc.)  Used .on('input') to detect typing

 Dynamically generated suggestion list using jQuery

 Added click handlers to populate search box with selected suggestion

**Code Snippet**

const menuSuggestions = [

'Latte', 'Cappuccino', 'Americano', 'Mocha', 'Espresso', 'Coffee', 'Milk', 'Hot', 'Cold', 'Sweet'

];

$('#menuSearch').on('input', function() {

const searchTerm = $(this).val().toLowerCase(); const matches = menuSuggestions.filter(item => item.toLowerCase().includes(searchTerm)

);

// Display suggestions

matches.forEach(match => { const li = $('<li>')

.addClass('list-group-item list-group-item-action')

.text(match)

.on('click', function() {

$('#menuSearch').val(match).trigger('keyup');

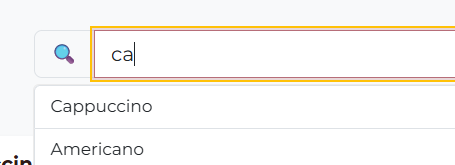
});

$('#searchSuggestions').append(li);

});

});

**Screenshots**

****

## Task 3: Search Highlighting

**Implemented by:** Balgyn

**Description**

Implemented text highlighting that marks all matching words on the page when a user searches. This makes it easier to visually identify search results.

**Implementation Details**

 Used regular expressions to find matching text

 Wrapped matches in <mark> tags with custom styling  Applied .html() method to modify content

 Added yellow highlight color for visibility

**Code Snippet**

const regex = new RegExp(`(${searchTerm})`, 'gi');

$('.card-body').each(function() { const element = $(this);

const html = element.html();

const newHtml = html.replace(regex, '<mark class="highlight">$1</mark>'); element.html(newHtml);

});

**CSS Styling**

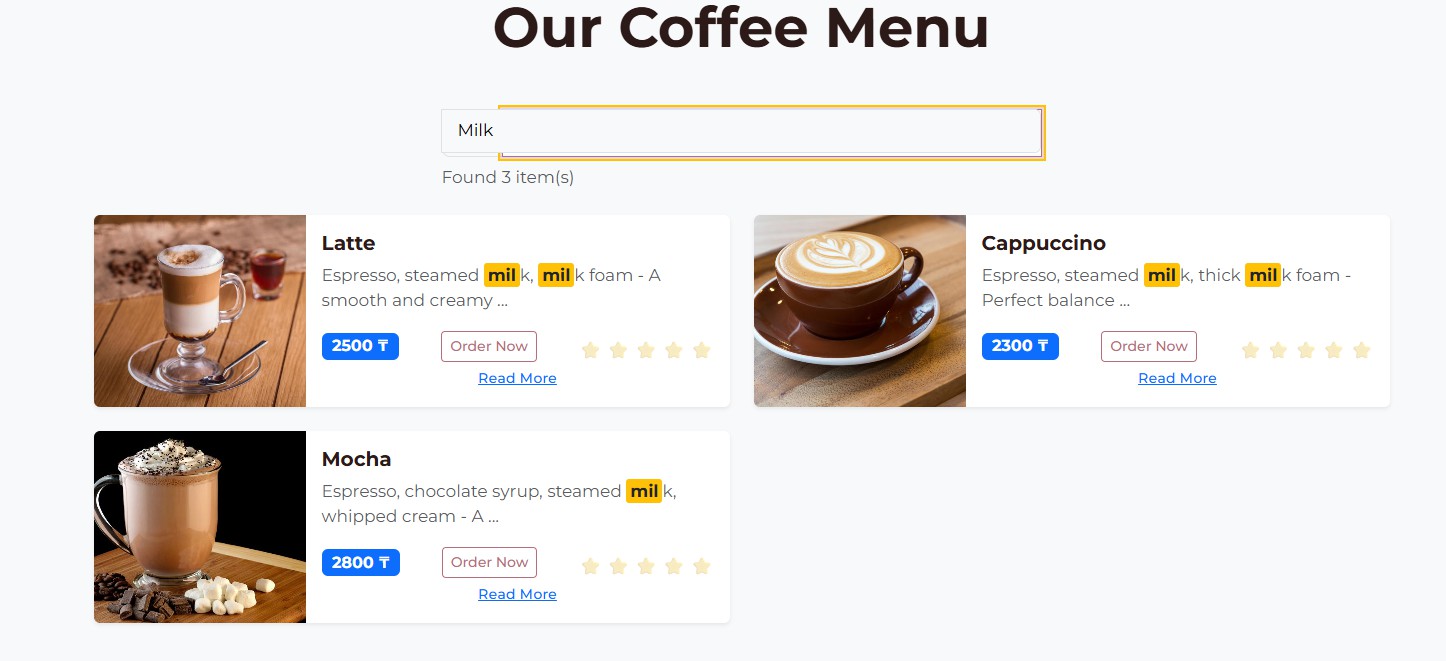
.highlight {

background-color: #ffc107; padding: 2px 4px;

border-radius: 3px; font-weight: bold;

}

**Screenshots**

****

# Part 2: UX Engagement Elements

## Task 4: Colorful and Stylized Scroll Progress Bar

**Implemented by:** Ayana

**Description**

Created a colorful progress bar at the top of the page that fills as users scroll down. This provides visual feedback about scroll position and adds a modern touch to the website.

**Implementation Details**

 Created fixed position div at top of page

 Used gradient colors (pink → yellow → green)

 Calculated scroll percentage using $(window).scrollTop()

 Updated width dynamically with .css() method

**Code Snippet**

$(window).on('scroll', function() {

const scrollTop = $(window).scrollTop();

const docHeight = $(document).height() - $(window).height(); const scrollPercent = (scrollTop / docHeight) \* 100;

$('#scrollProgress').css('width', scrollPercent + '%');

});

**Screenshots**

****

## Task 5: Animated Number Counter

**Implemented by:** Balgyn

**Description**

Implemented smooth counting animations for statistics (1000+ customers, 50+ varieties, 5+ years). Numbers animate from 0 to their target value when scrolled into view.

**Implementation Details**

 Created stats section with three counters

 Used .animate() method for smooth counting effect  Triggered animation when section becomes visible

 Duration set to 2000ms for smooth effect

**Code Snippet**

$({ counter: 0 }).animate({ counter: target }, { duration: 2000,

easing: 'swing', step: function() {

$this.text(Math.ceil(this.counter) + '+');

},

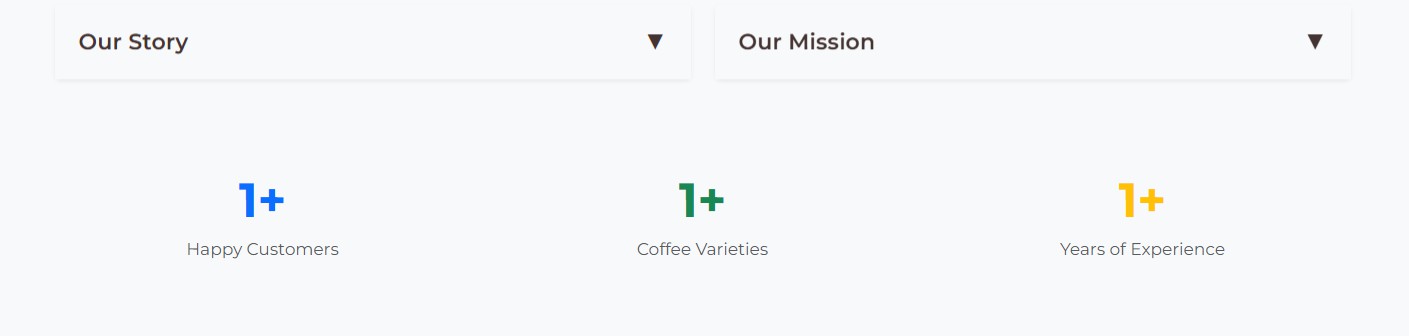
complete: function() {

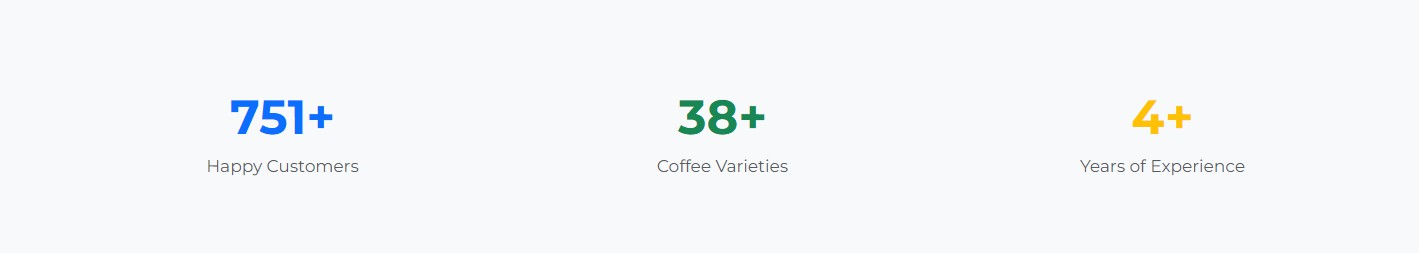
$this.text(target + '+');

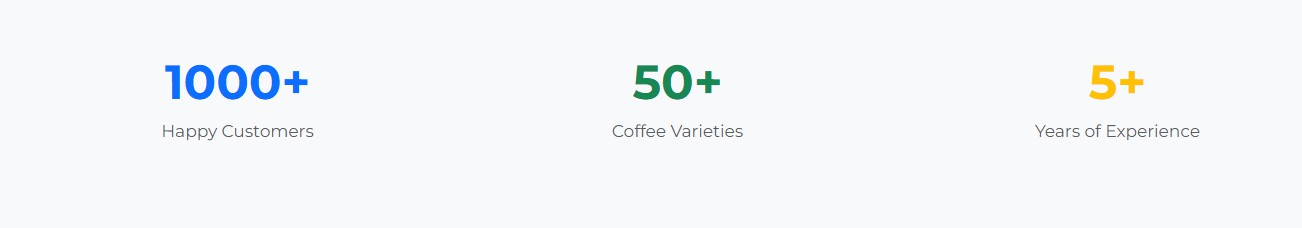
}

});

**Screenshots**

**Screenshot 13: Stats section before scrolling (counters at 0)**

**Screenshot 14: Stats section during animation (numbers counting up)**

**Screenshot 15: Stats section after animation (final values displayed)**

## Task 6: Loading Spinner on Submit

**Implemented by:** Ayana

**Description**

Created a search bar that filters menu items and gallery photos in real-time as the user types. This feature helps users quickly find specific coffee drinks or photos.

**Implementation Details**

 Used .on('keyup') event to detect user input

 Implemented .filter() method to show/hide matching items  Added search result counter showing number of matches

 Works on both menu.html and gallery.html pages

**Code Snippet**

$('#menuSearch').on('keyup', function() {

const searchTerm = $(this).val().toLowerCase(); let visibleCount = 0;

$('.col-lg-6.col-md-12').filter(function() {

const cardText = $(this).text().toLowerCase();

const matches = cardText.indexOf(searchTerm) > -1;

$(this).toggle(matches);

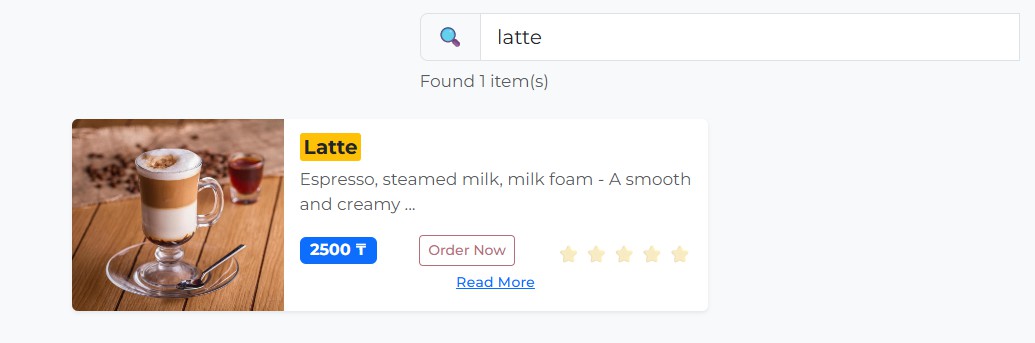
if (matches) visibleCount++; return true;

});

$('#searchResults').text(`Found ${visibleCount} item(s)`);

});

**Screenshots**

****

# Part 3: Web App Functionality Improvements

## Task 7: Notification System

**Implemented by:** Balgyn

**Description**

Created a toast notification system that displays temporary messages for user actions like "Item added to cart" or "Form submitted successfully". Notifications automatically fade out after 4 seconds.

**Implementation Details**

 Created fixed notification container in top-right corner  Different notification types: success, info, warning, error  Used .fadeOut() for smooth disappearing effect

 Positioned using position: fixed with z-index  Auto-dismiss with setTimeout()

**Code Snippet**

function showNotification(message, type = 'info') { const types = {

'success': 'alert-success', 'info': 'alert-info',

'warning': 'alert-warning', 'error': 'alert-danger'

};

const notification = $(`

<div class="alert ${types[type]} alert-dismissible fade show shadow-lg" role="alert">

${message}

<button type="button" class="btn-close" data-bs-dismiss="alert">

</button>

</div>

`);

$('#notificationContainer').append(notification); setTimeout(function() {

notification.fadeOut(500, function() {

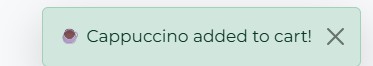
$(this).remove();

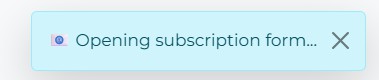
});

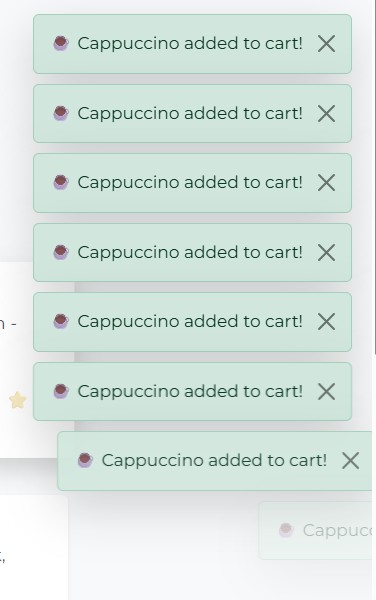
}, 4000);

}

**Screenshots**

**Screenshot 19: Success notification "Item added to cart"**

**Screenshot 20: Info notification "Opening subscription form..."**

**Screenshot 21: Multiple notifications stacked **

## Task 8: Copied to Clipboard Button

**Implemented by:** Ayana

**Description**

Added "Copy" buttons next to phone numbers and addresses in the footer. When clicked, the button copies text to clipboard, changes to a checkmark, shows a tooltip, and displays a notification.

**Implementation Details**

 Used navigator.clipboard.writeText() API  Added click handlers with .on('click')

 Visual feedback: button changes to green with checkmark  Toast notification confirms successful copy

 Button resets after 2 seconds

**Code Snippet**

$('.text-white.text-decoration-none.fs-5').each(function() { const phoneNumber = $(this).text();

const copyBtn = $(`

<button class="btn btn-sm btn-outline-light ms-2 copy-phone-btn" data-phone="${phoneNumber}">

📋 Copy

</button>

`);

$(this).after(copyBtn);

copyBtn.on('click', function() {

const phone = $(this).attr('data-phone');

navigator.clipboard.writeText(phone).then(() => {

$(this).html('✓ Copied!')

.removeClass('btn-outline-light')

.addClass('btn-success');

showNotification('📋 Phone number copied to clipboard!', 'success'); setTimeout(() => {

$(this).html('📋 Copy')

.removeClass('btn-success')

.addClass('btn-outline-light');

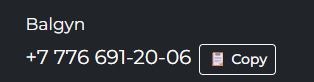
}, 2000);

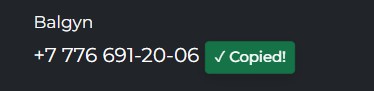
});

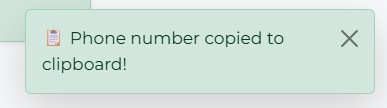
});

});

**Screenshots**

**Screenshot 22: Copy buttons next to phone numbers in footer**

**Screenshot 23: Button showing "**✓ **Copied!" after clicking**

**Screenshot 24: Clipboard notification confirming copy action**

## Task 9: Image Lazy Loading

**Implemented by:** Balgyn

**Description**

Implemented lazy loading for images throughout the website. Images only load when they come into the viewport, improving page load performance and reducing bandwidth usage.

**Implementation Details**

 Replaced src attribute with data-src for all images  Added placeholder SVG while images load

 Used $(window).on('scroll') to detect viewport position  Calculated image position with .offset().top

 Loaded images when within 200px of viewport

**Code Snippet**

$('img').not('.carousel-img').each(function() { const $img = $(this);

const src = $img.attr('src');

if (src && !$img.hasClass('lazy-loaded')) {

$img.attr('data-src', src)

.attr('src', 'data:image/svg+xml,...') // Placeholder

.addClass('lazy-image');

}

});

function lazyLoadImages() {

$('.lazy-image').each(function() { const $img = $(this);

const imgTop = $img.offset().top;

const viewportBottom = $(window).scrollTop() + $(window).height();

if (imgTop < viewportBottom + 200) {

const dataSrc = $img.attr('data-src');

$img.attr('src', dataSrc)

.removeClass('lazy-image')

.addClass('lazy-loaded')

.fadeIn(500);

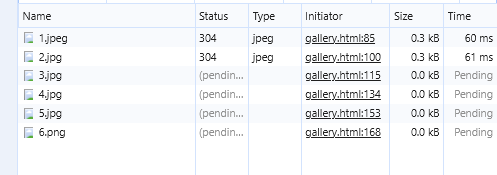
}

});

}

$(window).on('scroll', lazyLoadImages);

**Screenshots**

****

# Part 4: Deployment

## GitHub Repository

**Repository URL:** [[Web site]](https://aaituu.github.io/BalgynWEB/web7/)

# Conclusion

Assignment 7 successfully enhanced our Coffee PO website with powerful jQuery features. We implemented nine major functionalities that significantly improved user experience:

 **Search Features** made content discovery effortless

 **UX Elements** provided engaging visual feedback

 **Functional Improvements** enhanced overall usability

Both team members contributed equally, with Balgyn focusing on search and data manipulation features, while Ayana concentrated on UI/UX enhancements and user interactions.

Our website is now more interactive, performant, and user-friendly, demonstrating modern web development best practices.