```
document.addEventListener('DOMContentLoaded', function() {
    console.log('DOM fully loaded and parsed');

    // Initialize Mixpanel
    mixpanel.init("e0d9428c16e42da11c02d8ff82ab4378", {
        debug: true,
        track_pageview: true,
        persistence: "localStorage",
    });

    // Update data-tracker attributes for all menu items
    document.querySelectorAll('.menu-item').forEach(function(item) {
        var itemId = item.getAttribute('data-tracker');
        item.setAttribute('data-tracker', 'Menu Item Click\n' + itemId);
    });

    // Update data-tracker attributes for category items
    document.querySelectorAll('.menu-category').forEach(function(item) {
        var itemId = item.getAttribute('data-tracker');
        item.setAttribute('data-tracker', 'Menu Category Click\n' + itemId);
    });

    // Click event tracking
    $(document).ready(function() {
        $(document).on('click', '[data-tracker]', debounce(function(e) {
            var trackData = $(this).data('tracker');
            console.log('Track data:', trackData); // Debugging log
            if (!trackData) {
                return;
            }
            var tagData = ParseTagData.tagData(trackData);
            console.log('Parsed tag data:', tagData); // Debugging log
            if (!tagData.action || !tagData.label) {
                return;
            }
            Track.trackEvent('click', { action: tagData.action, label: tagData.label });
        }, 300)); // Adjust the debounce wait time as needed
    });

        var ParseTagData = {
            tagData: function(data) {
                var tmpData = data.split("\n");
                if (tmpData.length != 2) {
                    return "";
```

```
            }
            return {
                "action": tmpData[0],
                "label": tmpData[1]
            };
        }
    };

    var Track = {
        trackEvent: function(eventType, data) {
            console.log('Tracking event:', eventType, data); // Debugging log
            mixpanel.track(data.action, {
                label: data.label
            });
        }
    };
});

// Debounce function to limit the rate of function calls
function debounce(func, wait) {
    let timeout;
    return function(...args) {
        clearTimeout(timeout);
        timeout = setTimeout(() => func.apply(this, args), wait);
    };
}

// Customization data object
const customizationData = {};

// Function to save customization state
function saveCustomizationState(itemName, customizationData) {
    localStorage.setItem(itemName, JSON.stringify(customizationData));
}

// Function to load customization state
function loadCustomizationState(itemName) {
    const data = localStorage.getItem(itemName);
    return data ? JSON.parse(data) : null;
}

let selectedItemName = null;
let preselectedQuantity = '';
```

```
    // Function to hide all cart items initially
    function hideAllCartItems() {
        const cartItems = document.querySelectorAll('.cart-list .cart-item');
        cartItems.forEach(item => {
            item.style.display = 'none';
            item.classList.remove('focused');
            console.log('Hiding cart item:', item);
        });
    }

    // Function to hide all customization contents initially
    function hideAllCustomizationContents() {
        const customizationContentsTab1 =
document.querySelectorAll('.customization-list .customization-content');
        const customizationContentsTab2 = document.querySelectorAll('#drink-
selection .customization-list .customization-content');
        customizationContentsTab1.forEach(item => {
            item.style.display = 'none';
            console.log('Hiding customization content for Tab 1:', item);
        });
        customizationContentsTab2.forEach(item => {
            item.style.display = 'none';
            console.log('Hiding customization content for Tab 2:', item);
        });
    }

    // Function to hide all item customizations initially
    function hideAllItemCustomizations() {
        const itemCustomizationsTab1 = document.querySelectorAll('.item-
customization');
        const itemCustomizationsTab2 = document.querySelectorAll('#drink-
selection .item-customization');
        itemCustomizationsTab1.forEach(item => {
            item.style.display = 'none';
            console.log('Hiding item customization for Tab 1:', item);
        });
        itemCustomizationsTab2.forEach(item => {
            item.style.display = 'none';
            console.log('Hiding item customization for Tab 2:', item);
        });
    }

    // Function to show only the selected item in item-customization-list
```

```
    function showSelectedItemCustomization(itemName) {
        hideAllItemCustomizations();
        const selectedItemCustomizationTab1 = document.querySelector(`.item-
customization[data-cart-customization="${itemName}"]`);
        const selectedItemCustomizationTab2 = document.querySelector(`#drink-
selection .item-customization[data-cart-customization="${itemName}"]`);
        if (selectedItemCustomizationTab1) {
            selectedItemCustomizationTab1.style.display = 'block';
            console.log('Showing item customization for Tab 1:', itemName);
        } else {
            console.log('No item customization found for Tab 1:', itemName);
        }
        if (selectedItemCustomizationTab2) {
            selectedItemCustomizationTab2.style.display = 'block';
            console.log('Showing item customization for Tab 2:', itemName);
        } else {
            console.log('No item customization found for Tab 2:', itemName);
        }
    }

    // Function to move the most recently selected item to the bottom of the cart list
    function moveToBottom(cartItem) {
        const cartList = document.querySelector('.cart-list');
        cartList.appendChild(cartItem);
        console.log('Moved cart item to bottom:', cartItem);
    }

    // Function to increment the amount of cart items by a specific quantity
    function incrementCartItem(itemName, quantity = 1) {
        const correspondingCartItem = document.querySelector(`.cart-list .cart-
item[data-cart-item="${itemName}"]`);
        if (correspondingCartItem) {
            const amountElement = correspondingCartItem.querySelector('[data-cart-
amount]');
            let currentAmount = parseInt(amountElement.getAttribute('amount'), 10);
            currentAmount = isNaN(currentAmount) ? 0 : currentAmount;
            currentAmount += quantity;
            amountElement.setAttribute('amount', currentAmount);
            amountElement.textContent = currentAmount;
            console.log('Updated cart item amount:', currentAmount);
            updateSubCartItems(itemName, currentAmount); // Update sub-cart-item
amounts
            updateQuantityControl(itemName, currentAmount); // Update quantity
control amount for both tabs
```

```
            customizationData[itemName] = customizationData[itemName] || {};
            customizationData[itemName].amount = currentAmount;
            saveCustomizationState(itemName, customizationData[itemName]);
            updateCartItemPrice(itemName);
            updateCartSubtotal();
            return currentAmount;
        } else {
            console.log('No corresponding cart item found for data-cart-item:',
itemName);
            return null;
        }
    }

    // Function to decrement the amount of cart items
    function decrementCartItem(itemName) {
        const correspondingCartItem = document.querySelector(`.cart-list .cart-
item[data-cart-item="${itemName}"]`);
        if (correspondingCartItem) {
            const amountElement = correspondingCartItem.querySelector('[data-cart-
amount]');
            let currentAmount = parseInt(amountElement.getAttribute('amount'), 10);
            currentAmount = isNaN(currentAmount) ? 1 : currentAmount;
            currentAmount = Math.max(1, currentAmount - 1);
            amountElement.setAttribute('amount', currentAmount);
            amountElement.textContent = currentAmount;
            console.log('Updated cart item amount:', currentAmount);
            updateSubCartItems(itemName, currentAmount); // Update sub-cart-item
amounts
            updateQuantityControl(itemName, currentAmount); // Update quantity
control amount for both tabs
            customizationData[itemName] = customizationData[itemName] || {};
            customizationData[itemName].amount = currentAmount;
            saveCustomizationState(itemName, customizationData[itemName]);
            updateCartItemPrice(itemName);
            updateCartSubtotal();
            return currentAmount;
        } else {
            console.log('No corresponding cart item found for data-cart-item:',
itemName);
            return null;
        }
    }

    // Function to update sub-cart-item amounts
```

```
    function updateSubCartItems(itemName, currentAmount) {
        const subCartItems = document.querySelectorAll(`.sub-cart-item[data-cart-
item="${itemName}"] .order-item-quantity.is--small`);
        subCartItems.forEach(subCartItem => {
            subCartItem.textContent = currentAmount;
            console.log('Updated sub-cart-item amount to:', currentAmount);
        });
    }

    // Function to update the quantity-control amount field for the corresponding
customization-content
    function updateQuantityControl(itemName, currentAmount) {
        const quantityControlAmountElements =
document.querySelectorAll(`.customization-content[data-cart-customization-
content="${itemName}"] .quantity-control .amount`);
        quantityControlAmountElements.forEach(amountElement => {
            if (amountElement) {
                amountElement.setAttribute('amount', currentAmount);
                amountElement.textContent = currentAmount;
                console.log('Updated quantity control amount:', currentAmount);
            }
        });
    }

    // Function to update the price based on quantity
    function updateCartItemPrice(itemName, basePrice = null) {
        const correspondingCartItem = document.querySelector(`.cart-list .cart-
item[data-cart-item="${itemName}"]`);
        if (correspondingCartItem) {
            const amountElement = correspondingCartItem.querySelector('[data-cart-
amount]');
            const priceElement = correspondingCartItem.querySelector('[data-cart-
price]');
            if (amountElement && priceElement) {
                const price = basePrice !== null ? basePrice :
parseFloat(priceElement.getAttribute('data-cart-price'));
                const quantity = parseInt(amountElement.getAttribute('amount'), 10);
                console.log('Price:', price);
                console.log('Quantity:', quantity);
                if (!isNaN(price) && !isNaN(quantity)) {
                    const totalPrice = (price * quantity).toFixed(2);
                    priceElement.textContent = totalPrice;
                    console.log(`Updated price for ${itemName}: ${totalPrice}`);
                } else {
```

```
                console.log('Invalid price or quantity for', itemName);
            }
        } else {
            console.log('Amount or price element not found for', itemName);
        }
    } else {
        console.log('No corresponding cart item found for data-cart-item:',
itemName);
    }
}

// Function to update the cart subtotal
function updateCartSubtotal() {
    const cartItems = document.querySelectorAll('.cart-list .cart-item');
    let subtotal = 0;
    cartItems.forEach(item => {
        const priceElement = item.querySelector('[data-cart-price]');
        if (priceElement && item.style.display !== 'none') {
            const price = parseFloat(priceElement.textContent);
            if (!isNaN(price)) {
                subtotal += price;
            }
        }
    });
    const subtotalElement = document.getElementById('cart-subtotal');
    if (subtotalElement) {
        subtotalElement.textContent = subtotal.toFixed(2);
    }
    // Update the cart tax based on the new subtotal
    updateCartTax(subtotal);
}

// Function to update the cart tax
function updateCartTax(subtotal) {
    const taxRate = 0.10; // 10% tax rate
    const tax = subtotal * taxRate;
    const taxElement = document.getElementById('cart-tax');
    if (taxElement) {
        taxElement.textContent = tax.toFixed(2);
    }
}

// Function to update the menu-list grid columns
function updateMenuListGrid(isPanelOpen) {
```

```
        const menuList = document.querySelector('.menu-list');
        const screenWidth = window.innerWidth;
        let columns;
        if (screenWidth < 1920) {
            columns = isPanelOpen ? 2 : 3;
        } else {
            columns = isPanelOpen ? 3 : 4;
        }
        if (menuList) {
            menuList.style.gridTemplateColumns = `repeat(${columns}, 1fr)`;
            console.log(`Updated menu-list grid to ${columns} columns`);
        } else {
            console.log('Menu list not found');
        }
    }

    // Function to update the min-width of cart-customizations based on screen size
and panel state
    function updateCartCustomizationsWidth(isPanelOpen) {
        const cartCustomizations = document.querySelector('.cart-customizations');
        if (cartCustomizations) {
            const screenWidth = window.innerWidth;
            if (screenWidth < 1920) {
                cartCustomizations.style.minWidth = isPanelOpen ? '496px' : '248px';
            } else {
                cartCustomizations.style.minWidth = isPanelOpen ? '800px' : '400px';
            }
            console.log(`Updated cart-customizations min-width to $
{cartCustomizations.style.minWidth}`);
        } else {
            console.log('Cart customizations not found');
        }
    }

    // Function to update the customization panel
    function updateCustomizationPanel(itemName) {
        const customizationPanel = document.querySelector('.customization-panel');
        if (customizationPanel) {
            customizationPanel.style.display = 'flex';
            updateMenuListGrid(true);
            updateCartCustomizationsWidth(true);
            hideAllCustomizationContents();

            const correspondingCustomizationContentTab1 =
```

```
document.querySelector(`.customization-list .customization-content[data-cart-customization-content="${itemName}"]`);
        const correspondingCustomizationContentTab2 = document.querySelector(`#drink-selection .customization-list .customization-content[data-cart-customization-content="${itemName}"]`);

        if (correspondingCustomizationContentTab1) {
            correspondingCustomizationContentTab1.style.display = 'flex';
            console.log('Showing customization content for Tab 1:', itemName);
        } else {
            console.log('No corresponding customization content found for Tab 1:', itemName);
        }

        if (correspondingCustomizationContentTab2) {
            correspondingCustomizationContentTab2.style.display = 'flex';
            console.log('Showing customization content for Tab 2:', itemName);
        } else {
            console.log('No corresponding customization content found for Tab 2:', itemName);
        }

        // Retrieve the customization data
        const customization = customizationData[itemName] || loadCustomizationState(itemName) || {};
        const amount = customization.amount || 1;
        const customizationText = customization.customizationText || '';

        // Update the quantity control amount for both tabs
        const quantityControlTab1 = correspondingCustomizationContentTab1 ? correspondingCustomizationContentTab1.querySelector('.quantity-control') : null;
        const quantityControlTab2 = correspondingCustomizationContentTab2 ? correspondingCustomizationContentTab2.querySelector('.quantity-control') : null;

        if (quantityControlTab1) {
            const amountElement = quantityControlTab1.querySelector('.amount');
            if (amountElement) {
                amountElement.setAttribute('amount', amount);
                amountElement.textContent = amount;
                console.log('Updated quantity control amount for Tab 1:', amount);
            }
        }

        if (quantityControlTab2) {
```

```
            const amountElement = quantityControlTab2.querySelector('.amount');
            if (amountElement) {
                amountElement.setAttribute('amount', amount);
                amountElement.textContent = amount;
                console.log('Updated quantity control amount for Tab 2:', amount);
            }
        }

        // Update the customization text for both tabs
        const descriptionElementTab1 = correspondingCustomizationContentTab1 ?
correspondingCustomizationContentTab1.querySelector('.order-item-
description') : null;
        const descriptionElementTab2 =
correspondingCustomizationContentTab2 ?
correspondingCustomizationContentTab2.querySelector('.order-item-
description') : null;

        if (descriptionElementTab1) {
            descriptionElementTab1.textContent = customizationText;
            descriptionElementTab1.style.display = customizationText ? 'block' :
'none';
        }

        if (descriptionElementTab2) {
            descriptionElementTab2.textContent = customizationText;
            descriptionElementTab2.style.display = customizationText ? 'block' :
'none';
        }

        // Ensure the description remains visible when switching tabs
        if (descriptionElementTab1 && descriptionElementTab1.style.display ===
'block') {
            if (descriptionElementTab2) {
                descriptionElementTab2.style.display = 'block';
            }
        }

        // Update the state of customization ingredients for both tabs
        const ingredients = customization.ingredients || {};
        const customizationIngredientsTab1 =
correspondingCustomizationContentTab1 ?
correspondingCustomizationContentTab1.querySelectorAll('.customization-
ingredient') : [];
        const customizationIngredientsTab2 =
```

```
correspondingCustomizationContentTab2 ?
correspondingCustomizationContentTab2.querySelectorAll('.customization-
ingredient') : [];

        customizationIngredientsTab1.forEach(ingredient => {
            const ingredientName = ingredient.getAttribute('ingredient-name');
            if (ingredients[ingredientName]) {
                ingredient.classList.add('focused');
            } else {
                ingredient.classList.remove('focused');
            }
        });

        customizationIngredientsTab2.forEach(ingredient => {
            const ingredientName = ingredient.getAttribute('ingredient-name');
            if (ingredients[ingredientName]) {
                ingredient.classList.add('focused');
            } else {
                ingredient.classList.remove('focused');
            }
        });

        // Add event listeners to quantity control buttons for both tabs
        addQuantityControlListeners();
    } else {
        console.log('Customization panel not found');
    }
}

// Function to add event listeners to quantity control buttons
function addQuantityControlListeners() {
    const quantityAddButtons = document.querySelectorAll('#quantity-add');
    const quantitySubtractButtons = document.querySelectorAll('#quantity-
subtract');

    quantityAddButtons.forEach(button => {
        button.removeEventListener('click', handleQuantityAddClick); // Remove
existing listener
        button.addEventListener('click', handleQuantityAddClick); // Add new
listener
    });

    quantitySubtractButtons.forEach(button => {
        button.removeEventListener('click', handleQuantitySubtractClick); //
```

```
Remove existing listener
            button.addEventListener('click', handleQuantitySubtractClick); // Add new
listener
        });
    }

    // Event handler for quantity add button click
    const handleQuantityAddClick = debounce(function(event) {
        const itemName = event.target.closest('.customization-
content').getAttribute('data-cart-customization-content');
        incrementCartItem(itemName);
    }, 300);

    // Event handler for quantity subtract button click
    const handleQuantitySubtractClick = debounce(function(event) {
        const itemName = event.target.closest('.customization-
content').getAttribute('data-cart-customization-content');
        decrementCartItem(itemName);
    }, 300);

    // Object to store the tab state for each item
    const itemTabState = {};

    // Function to switch tabs
    function switchTab(tabIndex) {
        console.log(`Switching to tab index: ${tabIndex}`);
        const customizationPanel = document.querySelector('.customization-panel');
        if (customizationPanel) {
            const tabId = tabIndex === 1 ? 'drink-selection' : tabIndex === 2 ? 'meal-
review' : 'item-customization'; // Update with actual tab 3 ID
            customizationPanel.setAttribute('data-current', `Tab ${tabIndex + 1}`);
            console.log(`Set customization-panel data-current to: Tab ${tabIndex + 1}
`);
            const tabs = customizationPanel.querySelectorAll('.customization-step');
            tabs.forEach(tab => {
                tab.style.display = tab.id === tabId ? 'block' : 'none';
                console.log(`Tab ${tab.id} display: ${tab.style.display}`);
            });
        } else {
            console.log('Customization panel not found');
        }
    }

    // Event listener for the "make-meal" button
```

```
    document.querySelectorAll('#make-meal').forEach(button => {
        button.addEventListener('click', function() {
            const itemName = this.closest('.item-customization').getAttribute('data-
name');
            console.log(`"make-meal" button clicked for item: ${itemName}`);

            // Get the current amount of the item
            const correspondingCartItem = document.querySelector(`.cart-list .cart-
item[data-cart-item="${itemName}"]`);
            const amountElement = correspondingCartItem.querySelector('[data-cart-
amount]');
            const currentAmount = amountElement ?
parseInt(amountElement.getAttribute('amount'), 10) : 1;

            // Set the amount in the Tab 2 quantity-control
            const quantityControlAmountElementTab2 =
document.querySelector(`#drink-selection .customization-content[data-cart-
customization-content="${itemName}"] .quantity-control .amount`);
            if (quantityControlAmountElementTab2) {
                quantityControlAmountElementTab2.setAttribute('amount',
currentAmount);
                quantityControlAmountElementTab2.textContent = currentAmount;
                console.log('Updated quantity control amount for Tab 2:',
currentAmount);
            }

            // Show the meal-info section and set the amount for sub-cart-item
elements
            const mealInfoSection = correspondingCartItem.querySelector('.meal-
info');
            if (mealInfoSection) {
                mealInfoSection.style.display = 'flex';
                console.log('Displayed meal-info section for item:', itemName);

                // Set the amount for sub-cart-item elements
                const subCartItems = mealInfoSection.querySelectorAll('.sub-cart-
item .order-item-quantity.is--small');
                subCartItems.forEach(subCartItem => {
                    subCartItem.textContent = currentAmount;
                    console.log('Updated sub-cart-item amount to:', currentAmount);
                });

                // Show meal-side and drink-selection, hide meal-drink and side-
selection
```

```
            const mealSide = mealInfoSection.querySelector('#meal-side');
            const drinkSelection = mealInfoSection.querySelector('#drink-
selection');
            const mealDrink = mealInfoSection.querySelector('#meal-drink');
            const sideSelection = mealInfoSection.querySelector('#side-selection');
            if (mealSide) mealSide.style.display = 'flex';
            if (drinkSelection) drinkSelection.style.display = 'flex';
            if (mealDrink) mealDrink.style.display = 'none';
            if (sideSelection) sideSelection.style.display = 'none';
        }

        // Show the meal-tag element
        const mealTagElement = correspondingCartItem.querySelector('.cart-item-
info .item-content .order-item-price-container .meal-tag');
        if (mealTagElement) {
            mealTagElement.style.display = 'block';
            console.log('Displayed meal-tag element for item:', itemName);
        }

        itemTabState[itemName] = 1; // Store the tab state as tab 2 (index 1)
        switchTab(1); // Switch to tab 2 (index 1)
    });
});

// Function to handle item selection and restore tab state
function handleItemSelection(itemName) {
    const tabIndex = itemTabState[itemName] || 0; // Default to tab 1 (index 0)
    console.log(`Item selected: ${itemName}, restoring tab index: ${tabIndex}`);
    switchTab(tabIndex);
}

// Function to handle cart item click
function handleCartItemClick(itemName) {
    console.log('handleCartItemClick called with:', itemName);
    const customizationPanel = document.querySelector('.customization-panel');
    const correspondingCartItem = document.querySelector(`.cart-list .cart-
item[data-cart-item="${itemName}"]`);

    if (selectedItemName === itemName) {
        if (customizationPanel) {
            customizationPanel.style.display = 'none';
            updateMenuListGrid(false);
            updateCartCustomizationsWidth(false);
            console.log('Hiding customization panel');
```

```
            }
            correspondingCartItem.classList.remove('focused');
            selectedItemName = null;
        } else {
            selectedItemName = itemName;
            console.log('Selected item name set to:', selectedItemName);
            const cartItems = document.querySelectorAll('.cart-list .cart-item');
            cartItems.forEach(item => item.classList.remove('focused'));
            correspondingCartItem.classList.add('focused');
            // Update the customization panel with the stored customization data
            updateCustomizationPanel(itemName);
            showSelectedItemCustomization(itemName);

            // Restore the tab state for the selected item
            handleItemSelection(itemName);

            // Update the quantity-control amount field
            const amountElement = correspondingCartItem.querySelector('[data-cart-
amount]');
            const quantityControlAmountElement =
document.querySelector(`.customization-content[data-cart-customization-
content="${itemName}"] .quantity-control .amount`);
            if (amountElement && quantityControlAmountElement) {
                const currentAmount = amountElement.getAttribute('amount');
                quantityControlAmountElement.setAttribute('amount', currentAmount);
                quantityControlAmountElement.textContent = currentAmount;
                console.log('Updated quantity control amount:', currentAmount);
            }
        }
    }

    // Function to handle menu item click
    const handleMenuItemClick = debounce(function(itemName) {
        console.log('handleMenuItemClick called with:', itemName);
        let correspondingCartItem = document.querySelector(`.cart-item[data-cart-
item="${itemName}"]`);
        let quantityToAdd = preselectedQuantity ? parseInt(preselectedQuantity, 10) :
1;
        preselectedQuantity = ''; // Reset preselected quantity after adding to cart
        updatePreselectedQuantityDisplay();

        if (correspondingCartItem && correspondingCartItem.style.display !== 'none')
{
            // If the cart item is already visible, increment the amount
```

```
            const newAmount = incrementCartItem(itemName, quantityToAdd);
            // Update the quantity-control amount field
            const quantityControlAmountElement =
document.querySelector('.quantity-control .amount');
            if (quantityControlAmountElement) {
                quantityControlAmountElement.setAttribute('amount', newAmount);
                quantityControlAmountElement.textContent = newAmount;
                console.log('Updated quantity control amount:', newAmount);
            }
        } else {
            if (!correspondingCartItem) {
                // Create a new cart item if it doesn't exist
                correspondingCartItem = document.createElement('div');
                correspondingCartItem.classList.add('cart-item');
                correspondingCartItem.setAttribute('data-cart-item', itemName);
                correspondingCartItem.innerHTML = `
                    <div class="amount" data-cart-amount="0">0</div>
                    <div class="order-item-description is--cart-description" data-cart-
description="item-description" style="display: none;">-</div>
                    <!-- Add other necessary elements for the cart item -->
                `;
                document.querySelector('.cart-
list').appendChild(correspondingCartItem);
                console.log('Added new cart item:', correspondingCartItem);
            } else {
                // Reset the cart item if it already exists but was hidden
                correspondingCartItem.style.display = 'flex';
                const amountElement = correspondingCartItem.querySelector('[data-
cart-amount]');
                if (amountElement) {
                    amountElement.setAttribute('amount', 0);
                    amountElement.textContent = 0;
                }
                const cartItemDescription =
correspondingCartItem.querySelector('[data-cart-description]');
                if (cartItemDescription) {
                    cartItemDescription.textContent = '-';
                    cartItemDescription.style.display = 'none';
                }
            }
            const cartItems = document.querySelectorAll('.cart-list .cart-item');
            cartItems.forEach(item => item.classList.remove('focused'));
            correspondingCartItem.classList.add('focused');
            console.log('Focused cart item:', correspondingCartItem);
```

```
            const newAmount = incrementCartItem(itemName, quantityToAdd);
            // Update the quantity-control amount field
            const quantityControlAmountElement =
document.querySelector('.quantity-control .amount');
            if (quantityControlAmountElement) {
                quantityControlAmountElement.setAttribute('amount', newAmount);
                quantityControlAmountElement.textContent = newAmount;
                console.log('Updated quantity control amount:', newAmount);
            }
            updateCustomizationPanel(itemName,
parseInt(correspondingCartItem.querySelector('[data-cart-
amount]').getAttribute('amount'), 10));
            updateCartItemPrice(itemName);
            showSelectedItemCustomization(itemName);
            selectedItemName = itemName;
            console.log('Selected item name set to:', selectedItemName);
            // Restore the tab state for the selected item
            handleItemSelection(itemName);
        }
    }, 300);

    // Add click event listener to each menu item
    function addClickListeners() {
        const menuItems = document.querySelectorAll('.menu-list .menu-item');
        if (menuItems.length === 0) {
            console.log('No menu items found');
        } else {
            menuItems.forEach(item => {
                item.addEventListener('click', function() {
                    const itemName = this.getAttribute('data-name');
                    console.log('Clicked menu item with data-name:', itemName);
                    handleMenuItemClick(itemName);
                });
            });
        }
        const cartItems = document.querySelectorAll('.cart-list .cart-item');
        if (cartItems.length === 0) {
            console.log('No cart items found');
        } else {
            cartItems.forEach(item => {
                item.addEventListener('click', function() {
                    const itemName = this.getAttribute('data-name');
                    console.log('Clicked cart item with data-name:', itemName);
                    handleCartItemClick(itemName);
```

```
            });
        });
    }
}

// Initial call to add click listeners and hide all cart items and customization
contents
hideAllCartItems();
hideAllCustomizationContents();
hideAllItemCustomizations();
addClickListeners();

// Function to handle menu category click
function handleMenuCategoryClick(event) {
    const target = event.target.closest('.menu-category');
    if (!target) return;
    const categoryId = target.id.replace('menu-category-', 'category-');
    if (!categoryId) {
        console.error('Category ID not found.');
        return;
    }
    const categorySection = document.getElementById(categoryId);
    if (categorySection) {
        categorySection.scrollIntoView({ behavior: 'smooth' });
    } else {
        console.error('Category section with ID', categoryId, 'not found.');
    }
}

// Add click event listeners to all menu-category elements
document.querySelectorAll('.menu-category').forEach(item => {
    item.addEventListener('click', handleMenuCategoryClick);
});

// Function to update the preselected quantity display
function updatePreselectedQuantityDisplay() {
    const preselectedQuantityElement = document.getElementById('preselected-
quantity');
    const clearButton = document.getElementById('quantity-clear');
    if (preselectedQuantityElement) {
        preselectedQuantityElement.textContent = preselectedQuantity || '-';
        if (preselectedQuantity) {
            clearButton.style.display = 'block';
        } else {
```

```
            clearButton.style.display = 'none';
        }
    } else {
        console.error('Element with ID "preselected-quantity" not found.');
    }
}

// Function to handle quantity number click
function handleQuantityNumberClick(event) {
    const target = event.target.closest('.quantity-number');
    if (!target) {
        console.error('Quantity number element not found.');
        return;
    }
    const number = target.id.split('-').pop();
    console.log('Quantity number clicked:', number); // Debugging log
    if (number === '0' && preselectedQuantity === '') {
        console.log('Cannot add 0 as the first digit');
        return; // Do not allow 0 as the first digit
    }
    if (preselectedQuantity.length < 2) {
        preselectedQuantity += number;
    } else {
        preselectedQuantity = number;
    }
    console.log('Updated preselectedQuantity:', preselectedQuantity); //
Debugging log
    updatePreselectedQuantityDisplay();
}

// Add click event listeners to quantity numbers
document.querySelectorAll('.quantity-number').forEach(item => {
    item.addEventListener('click', handleQuantityNumberClick);
});

// Function to handle clear button click
function handleClearButtonClick() {
    preselectedQuantity = '';
    updatePreselectedQuantityDisplay();
}

// Add click event listener to clear button
const clearButton = document.getElementById('quantity-clear');
if (clearButton) {
```

```
        clearButton.addEventListener('click', handleClearButtonClick);
    } else {
        console.error('Element with ID "quantity-clear" not found.');
    }

    // Use MutationObserver to detect when new menu items are added to the DOM
    const observer = new MutationObserver(addClickListeners);
    const config = { childList: true, subtree: true };
    const menuList = document.querySelector('.menu-list');
    if (menuList) {
        observer.observe(menuList, config);
    } else {
        console.log('Menu list not found');
    }

    // Initial call to update the cart subtotal and tax
    updateCartSubtotal();

    // Function to display the current date
    function formatDate(date) {
        const options = { weekday: 'short', year: 'numeric', month: 'short', day: '2-
digit' };
        return date.toLocaleDateString('en-US', options);
    }

    const currentDate = new Date();
    const formattedDate = formatDate(currentDate);
    const dateElement = document.getElementById('current-date');
    if (dateElement) {
        dateElement.textContent = formattedDate;
    } else {
        console.error('Element with ID "current-date" not found.');
    }

    // Function to display the current time
    function formatTime(date) {
        return date.toTimeString().split(' ')[0]; // Extracts the time portion in
HH:MM:SS format
    }

    const currentTime = new Date();
    const formattedTime = formatTime(currentTime);
    const timeElement = document.getElementById('current-time');
    if (timeElement) {
```

```
        timeElement.textContent = formattedTime;
    } else {
        console.error('Element with ID "current-time" not found.');
    }

    // Function to update the time every second
    function updateTime() {
        const currentTime = new Date();
        const formattedTime = formatTime(currentTime);
        const timeElement = document.getElementById('current-time');
        if (timeElement) {
            timeElement.textContent = formattedTime;
        } else {
            console.error('Element with ID "current-time" not found.');
        }
    }

    // Initial call to display the time immediately
    updateTime();

    // Set interval to update the time every second
    setInterval(updateTime, 1000);

    // Function to set the default focused size control button
    function setDefaultSizeControl() {
        const defaultButton = document.getElementById('size-control-medium');
        if (defaultButton) {
            defaultButton.classList.add('focused');
        }
    }

    // Function to update size text blocks
    function updateSizeTextBlocks(size, cartItem) {
        const mealItemSizeElement = cartItem.querySelector('#side-item-size[data-
size="meal-item-size"]');
        const mealSideSizeElement = cartItem.querySelector('#meal-side-size[data-
size="cart-item-size"][data-side-size="meal-side-size"]');
        const mealDrinkSizeElement = cartItem.querySelector('#meal-drink-
size[data-size="cart-item-size"][data-drink-size="meal-drink-size"]');

        if (mealItemSizeElement) {
            mealItemSizeElement.textContent = size;
            console.log('Updated meal-item-size to:', size);
        }
```

```
        if (mealSideSizeElement) {
            mealSideSizeElement.textContent = size;
            console.log('Updated meal-side-size to:', size);
        }
        if (mealDrinkSizeElement) {
            mealDrinkSizeElement.textContent = size;
            console.log('Updated meal-drink-size to:', size);
        }
    }

    // Function to handle size control button click
    function handleSizeControlClick(event) {
        const sizeControlButton = event.target;
        const size = sizeControlButton.getAttribute('data-size');
        const correspondingCartItem = document.querySelector(`.cart-item[data-
cart-item="${selectedItemName}"]`);

        if (!correspondingCartItem) return;

        // Update size control button states
        const sizeControlContainer = sizeControlButton.closest('.size-control-
container');
        if (!sizeControlContainer) return;

        const sizeControlButtons =
sizeControlContainer.querySelectorAll('.button.is--size-control');
        sizeControlButtons.forEach(button => button.classList.remove('focused'));
        sizeControlButton.classList.add('focused');

        // Update text elements based on selected size
        updateSizeTextBlocks(size, correspondingCartItem);
    }

    // Function to add event listeners to size control buttons
    function addSizeControlListeners() {
        const sizeControlButtons = document.querySelectorAll('.size-
control .button.is--size-control');
        sizeControlButtons.forEach(button => {
            button.removeEventListener('click', handleSizeControlClick); // Remove
existing listener
            button.addEventListener('click', handleSizeControlClick); // Add new
listener
        });
    }
```

```
    // Function to set the default focused size control button
    function setDefaultSizeControl() {
        const defaultButton = document.getElementById('size-control-medium');
        if (defaultButton) {
            defaultButton.classList.add('focused');
        }
    }

    // Initial setup for size control buttons
    setDefaultSizeControl();
    addSizeControlListeners();

    // Function to add event listeners to customization ingredients
    function addCustomizationIngredientListeners() {
        document.querySelectorAll('.customization-ingredient').forEach(item => {
            item.addEventListener('click', function() {
                const ingredientName = this.getAttribute('ingredient-name');
                const customizationType = this.closest('.customization-ingredient-
list').id.split('-').pop().toUpperCase();
                const customizationText = customizationType === 'REQUIRED' ?
ingredientName : `${customizationType} ${ingredientName}`;
                updateItemDescription(customizationText);
            });
        });
    }

    // Function to handle customization ingredient click
    function handleCustomizationIngredientClick(event) {
        const target = event.target.closest('.customization-ingredient');
        if (!target) return;
        const ingredientName = target.getAttribute('ingredient-name');
        const customizationType = target.closest('.customization-ingredient-
list').id.split('-').pop().toUpperCase();
        const customizationText = customizationType === 'REQUIRED' ?
ingredientName : `${customizationType} ${ingredientName}`;
        console.log('Customization ingredient clicked:', customizationText);
        // Toggle the focused class on the clicked element
        target.classList.toggle('focused');
        // Save the state of the customization ingredient for the selected item
        customizationData[selectedItemName] =
customizationData[selectedItemName] || {};
        customizationData[selectedItemName].ingredients =
customizationData[selectedItemName].ingredients || {};
```

```
        customizationData[selectedItemName].ingredients[ingredientName] =
target.classList.contains('focused');
        saveCustomizationState(selectedItemName,
customizationData[selectedItemName]);
        // Check if the quantity of the selected item is greater than 1
        const correspondingCartItem = document.querySelector(`.cart-item[data-
cart-item="${selectedItemName}"]`);
        const amountElement = correspondingCartItem.querySelector('[data-cart-
amount]');
        let currentAmount = parseInt(amountElement.getAttribute('amount'), 10);
        if (currentAmount > 1) {
            // Calculate the base price of the item
            const priceElement = correspondingCartItem.querySelector('[data-cart-
price]');
            const basePrice = parseFloat(priceElement.textContent) / currentAmount;
            // Decrement the quantity of the original item
            currentAmount -= 1;
            amountElement.setAttribute('amount', currentAmount);
            amountElement.textContent = currentAmount;
            // Update the price of the original item
            updateCartItemPrice(selectedItemName, basePrice);
            // Clone the existing cart item and update it with the customization
            cloneAndCustomizeCartItem(correspondingCartItem, customizationText,
basePrice);
        } else {
            // Update the item description
            updateItemDescription(customizationText);
        }
    }

    // Function to clone and customize the cart item
    function cloneAndCustomizeCartItem(originalCartItem, customizationText,
basePrice) {
        const cartList = document.querySelector('.cart-list');
        const newCartItem = originalCartItem.cloneNode(true);
        const uniqueId = `${originalCartItem.getAttribute('data-cart-item')}-$
{Date.now()}`;
        newCartItem.setAttribute('data-cart-item', uniqueId);
        // Ensure the meal-item has the correct data-meal-item attribute
        const originalMealItem = document.querySelector(`.meal-item[data-meal-
item="${originalCartItem.getAttribute('data-cart-item')}"]`);
        if (originalMealItem) {
            const newMealItem = originalMealItem.cloneNode(true);
            newMealItem.setAttribute('data-meal-item', uniqueId);
```

```
            const customizationContent = document.querySelector(`.customization-
content[data-cart-customization-content="${uniqueId}"] .meal-item-container`);
            if (customizationContent) {
                customizationContent.appendChild(newMealItem);
            }
        }
        // Update the cloned item with the customization details
        const amountElement = newCartItem.querySelector('[data-cart-amount]');
        if (amountElement) {
            amountElement.setAttribute('amount', 1);
            amountElement.textContent = '1';
        }
        const descriptionElement = newCartItem.querySelector('[data-cart-
description]');
        if (descriptionElement) {
            descriptionElement.textContent = customizationText;
            descriptionElement.style.display = 'block';
        }
        const priceElement = newCartItem.querySelector('[data-cart-price]');
        if (priceElement) {
            priceElement.textContent = basePrice.toFixed(2);
        }
        // Ensure the new cart item is visible and clickable
        newCartItem.style.display = 'flex';
        newCartItem.classList.add('focused');
        newCartItem.addEventListener('click', function() {
            handleCartItemClick(uniqueId);
        });
        cartList.appendChild(newCartItem);
        // Update the quantity-control amount field
        const quantityControlAmountElement =
document.querySelector(`.customization-content[data-cart-customization-
content="${uniqueId}"] .quantity-control .amount`);
        if (quantityControlAmountElement) {
            quantityControlAmountElement.setAttribute('amount', 1);
            quantityControlAmountElement.textContent = '1';
        }
        // Clone the corresponding item-customization element
        const originalItemCustomization = document.querySelector(`.item-
customization[data-cart-customization="${originalCartItem.getAttribute('data-
cart-item')}"]`);
        if (originalItemCustomization) {
            const newItemCustomization = originalItemCustomization.cloneNode(true);
            newItemCustomization.setAttribute('data-cart-customization', uniqueId);
```

```
            document.querySelector('.item-customization-
list').appendChild(newItemCustomization);
        }
        // Clone the corresponding customization-content element
        const originalCustomizationContent =
document.querySelector(`.customization-content[data-cart-customization-
content="${originalCartItem.getAttribute('data-cart-item')}"]`);
        if (originalCustomizationContent) {
            const newCustomizationContent =
originalCustomizationContent.cloneNode(true);
            newCustomizationContent.setAttribute('data-cart-customization-content',
uniqueId);
            document.querySelector('.customization-
list').appendChild(newCustomizationContent);
        }
        // Reset customization states for the original item
        const originalCustomizationIngredients =
document.querySelectorAll(`.customization-content[data-cart-customization-
content="${originalCartItem.getAttribute('data-cart-item')}"] .customization-
ingredient`);
        originalCustomizationIngredients.forEach(ingredient => {
            ingredient.classList.remove('focused');
        });

        // Update the cart subtotal and tax
        updateCartSubtotal();

        // Select the new cart item
        const cartItems = document.querySelectorAll('.cart-list .cart-item');
        cartItems.forEach(item => item.classList.remove('focused'));
        newCartItem.classList.add('focused');
        selectedItemName = uniqueId;

        // Store the customization data for the new item
        customizationData[uniqueId] = { customizationText: customizationText,
amount: 1, ingredients: {} };
        saveCustomizationState(uniqueId, customizationData[uniqueId]);

        // Update the customization panel for the new item
        updateCustomizationPanel(uniqueId);

        // Ensure the meal-item description is updated
        const mealItemDescription = document.querySelector(`.meal-item[data-
meal-item="${uniqueId}"] .order-item-description`);
```

```
        if (mealItemDescription) {
            mealItemDescription.textContent = customizationText;
            mealItemDescription.style.display = 'block';
        }
    }

    // Function to update item description based on customization ingredient click
    function updateItemDescription(customizationText) {
        console.log('updateItemDescription called with:', customizationText);
        if (!selectedItemName) {
            console.log('No selected item name');
            return;
        }
        const cartItemDescription = document.querySelector(`.cart-item[data-cart-item="${selectedItemName}"] .order-item-description.is--cart-description`);
        if (cartItemDescription) {
            let currentDescription = cartItemDescription.textContent.trim();
            if (currentDescription === '-') currentDescription = '';
            if (currentDescription.includes(customizationText)) {
                currentDescription = currentDescription.replace(customizationText, '').replace(/,\s*,/g, ',').replace(/^,|,$/g, '').trim();
            } else {
                currentDescription = currentDescription ? `${currentDescription}, ${customizationText}` : customizationText;
            }
            cartItemDescription.textContent = currentDescription || '-';
            console.log('Updated cart item description:', currentDescription);

            // Show the cart item description if it has customizations
            cartItemDescription.style.display = currentDescription ? 'block' : 'none';

            // Update the meal-item description to match the cart-item description for both tabs
            const mealItemDescriptionTab1 = document.querySelector(`.meal-item[data-meal-item="${selectedItemName}"] .order-item-description`);
            const mealItemDescriptionTab2 = document.querySelector(`#drink-selection .meal-item[data-meal-item="${selectedItemName}"] .order-item-description`);

            if (mealItemDescriptionTab1) {
                mealItemDescriptionTab1.textContent = currentDescription;
                mealItemDescriptionTab1.style.display = 'block';
                console.log('Updated meal item description for Tab 1 to match cart item description:', currentDescription);
```

```
            } else {
                console.log('Meal item description element not found for Tab 1:',
selectedItemName);
            }

            if (mealItemDescriptionTab2) {
                mealItemDescriptionTab2.textContent = currentDescription;
                mealItemDescriptionTab2.style.display = 'block';
                console.log('Updated meal item description for Tab 2 to match cart item
description:', currentDescription);
            } else {
                console.log('Meal item description element not found for Tab 2:',
selectedItemName);
            }
        } else {
            console.log('Cart item description element not found for:',
selectedItemName);
        }
    }

    // Function to handle drink selection click
    function handleDrinkSelectionClick(event) {
        const panelDrinkItem = event.target.closest('#panel-drink-item');
        if (!panelDrinkItem) return;

        const drinkName = panelDrinkItem.getAttribute('drink-name');
        if (!drinkName) return;

        const correspondingCartItem = document.querySelector(`.cart-item[data-
cart-item="${selectedItemName}"]`);
        if (!correspondingCartItem) return;

        // Hide the drink-selection element
        const drinkSelectionElement = correspondingCartItem.querySelector('#drink-
selection');
        if (drinkSelectionElement) {
            drinkSelectionElement.style.display = 'none';
            console.log('Hid drink-selection element for item:', selectedItemName);
        }

        // Display the meal-drink element
        const mealDrinkElement = correspondingCartItem.querySelector('#meal-
drink');
        if (mealDrinkElement) {
```

```
            mealDrinkElement.style.display = 'flex';
            console.log('Displayed meal-drink element for item:', selectedItemName);
        }

        // Update the meal-drink-item text
        const mealDrinkItemElement = correspondingCartItem.querySelector('#meal-
drink-item');
        if (mealDrinkItemElement) {
            mealDrinkItemElement.textContent = drinkName;
            mealDrinkItemElement.setAttribute('data-meal-drink-title', drinkName);
            console.log('Updated meal-drink-item text to:', drinkName);
        }

        // Switch to tab 3
        switchTab(2);
    }

    // Function to add event listeners to drink selection items
    function addDrinkSelectionListeners() {
        const panelDrinkItems = document.querySelectorAll('#panel-drink-
item.customization-ingredient.is--drink-selection');
        panelDrinkItems.forEach(item => {
            item.removeEventListener('click', handleDrinkSelectionClick); // Remove
existing listener
            item.addEventListener('click', handleDrinkSelectionClick); // Add new
listener
        });
    }

    // Function to remove the selected item from the cart and reset customizations
    function removeSelectedItem() {
        if (!selectedItemName) {
            console.log('No selected item to remove');
            return;
        }

        // Find the selected cart item
        const cartItem = document.querySelector(`.cart-item[data-name="$
{selectedItemName}"]`);
        if (cartItem) {
            // Hide the cart item instead of removing it
            cartItem.style.display = 'none';
            console.log('Hid cart item:', selectedItemName);
```

```
        // Reset the amount and description
        const amountElement = cartItem.querySelector('.amount');
        if (amountElement) {
            amountElement.setAttribute('amount', 0);
            amountElement.textContent = 0;
        }
        const cartItemDescription = cartItem.querySelector('.order-item-
description.is--cart-description');
        if (cartItemDescription) {
            cartItemDescription.textContent = '-';
            cartItemDescription.style.display = 'none';
        }

        // Reset customizations
        const mealItemDescription = document.querySelector(`.meal-item[data-
name="${selectedItemName}"] .order-item-description`);
        if (mealItemDescription) {
            mealItemDescription.textContent = '-';
        }
        const mealItemAmountElement = document.querySelector(`.meal-
item[data-name="${selectedItemName}"] .amount`);
        if (mealItemAmountElement) {
            mealItemAmountElement.setAttribute('amount', 0);
            mealItemAmountElement.textContent = 0;
        }

        // Reset customization ingredients
        const customizationIngredients =
document.querySelectorAll('.customization-ingredient.focused');
        customizationIngredients.forEach(ingredient => {
            ingredient.classList.remove('focused');
        });

        // Close the customization panel
        const customizationPanel = document.querySelector('.customization-
panel');
        if (customizationPanel) {
            customizationPanel.style.display = 'none';
            updateMenuListGrid(false);
            updateCartCustomizationsWidth(false);
            console.log('Closed customization panel');
        }
```

```
            // Hide the meal-info section
            const mealInfoSection = cartItem.querySelector('.meal-info');
            if (mealInfoSection) {
                mealInfoSection.style.display = 'none';
                console.log('Hid meal-info section for item:', selectedItemName);
            }

            // Hide the meal-tag element
            const mealTagElement = cartItem.querySelector('.cart-item-info .item-
content .order-item-price-container .meal-tag');
            if (mealTagElement) {
                mealTagElement.style.display = 'none';
                console.log('Hid meal-tag element for item:', selectedItemName);
            }

            // Reset the amount for sub-cart-item elements
            const subCartItems = mealInfoSection ?
mealInfoSection.querySelectorAll('.sub-cart-item .order-item-quantity.is--small') :
[];
            subCartItems.forEach(subCartItem => {
                subCartItem.textContent = 0;
                console.log('Reset sub-cart-item amount to 0');
            });

            // Clear the selected item name
            selectedItemName = null;

            // Reset the tab back to 1
            switchTab(0);

            // Clear the tab state for the removed item
            delete itemTabState[cartItem.getAttribute('data-cart-item')];
        } else {
            console.log('Cart item not found for:', selectedItemName);
        }
    }

    // Event delegation for dynamically added items
    document.addEventListener('click', function(event) {
        if (event.target.closest('.customization-ingredient')) {
            handleCustomizationIngredientClick(event);
        }
        if (event.target.closest('#remove-button')) {
```

```
        removeSelectedItem();
    }
});

// Ensure content is fully loaded before attaching event listeners
window.addEventListener('DOMContentLoaded', function() {
    jQuery(function() {
        // Content is loaded, event listeners are already attached via delegation
        addCustomizationIngredientListeners();
        addQuantityControlListeners();
        addDrinkSelectionListeners();
        setDefaultSizeControl();
        addSizeControlListeners();
    });
});
```