Wadau Hotel Reservation System Code Documentation

Overview

The Wadau Hotel Reservation System is a web-based application that allows customers to browse menu items, add them to a tray), checkout, provide their reservation details, pay and subscribe to the hotel. The system consists of several interconnected HTML pages and JavaScript files that work together to provide a seamless user experience.

File Structure and Flow

Core Pages

Html files

* index.html - Main landing page with featured specials and menu categories.
* category.html - Displays items within a selected category or subcategory.
* mytray.html - Shows items in the user's cart/tray with checkout functionality.
* reservation.html - Collects guest information and payment preference.
* payment.html - Handles payment method selection and processing.
* subscription.html - Confirmation page with subscription option.

JavaScript Files

* main.js - Handles the homepage functionality.
* category-page.js - Manages category and item display logic.
* mytray.js - Controls the cart/tray functionality.
* cart.js - Core cart operations and storage.
* menu-data.js - Contains all menu item data.
* reservation.js - Form handling for reservation details.
* payment.html (includes embedded JS) - Payment processing logic.
* subscription.js - Subscription management.

css files

* reservation.css - Shared styles for checkout flow.
* styles.css- sytling of all pages.

Detailed File Documentation

1. index.html

The main landing page that features:

* Restaurant logo and branding.
* "My Tray" button with item count indicator.
* "Our Special" section (desktop and mobile versions).
* Main menu categories (Platters, Main Course, Snacks, Beverages).
* Footer- hotel number

Key Features:

* Responsive design for desktop and mobile.
* Dynamic loading of special items.
* Navigation to category pages.
* Cart indicator updates.

2. category.html

Displays menu items based on selected category with:

* Back button navigation.
* Item carousel for featured items.
* Detailed item view on mobile.
* Quantity selection and add-to-tray functionality.

Key Features:

* Hierarchical category navigation (category → subcategory → sub-subcategory).
* Carousel display for featured items.
* Mobile-optimized item detail view.
* Real-time cart updates.

3. mytray.html

The cart interface showing:

* List of selected items.
* Quantity adjustment controls.
* Item removal.
* Total price calculation.
* Checkout button.

Key Features:

* Different layouts for desktop and mobile.
* Item quantity modification.
* Total price calculation.
* Checkout initiation.

4. main.js

Handles core homepage functionality:

Key Functions:

* updateSpecialSections() - Loads and displays featured/special items for home page.
* initializePage() - Sets up page elements and checks for required data.
* goToCategory(), goToMyTray(), goToThisWeekSpecial() - Navigation handlers.
* updateCartIndicator() - Updates the tray item count display.

Key Features:

* Special items display management.
* Navigation control.
* Cart indicator updates.

5. category-page.js

Core Logic & Functions

This file handles category navigation, item display, and cart interactions.

Code explanations.

1.Global declarations

* Cart System : Manages shopping cart state and persistence using localStorage. Provides all cart operations.

window.cart = {

items: [], // Array of cart items loaded from localStorage

// Core Operations:

addItem(), // Adds new item or updates existing item quantity

removeItem(), // Removes item from cart

updateQuantity(), // Updates quantity for specific item

getTotal(), // Calculates total cart value

getItemCount(), // Returns total number of items

saveToStorage(), // Persists cart to localStorage

dispatchUpdate(), // Triggers cart update event

clearCart() // Empties cart completely

}

* Special Items Configuration : Defines featured/highlighted items for specific categories with complete display information.

window.SPECIAL\_ITEMS = {

"category-id": {

id: "unique-id",

name: "Display Name",

description: "Item description",

image: "path/to/image.png"

}

}

* State Variables : Manages UI state for carousel and mobile views.

let currentCarouselIndex = 0; // Tracks current carousel position

let carouselItems = []; // Stores items for carousel display

let currentMobileItem = null; // Tracks currently displayed mobile item

2. CORE FUNCTIONALITY

Page Initialization

* function initializePage()

Main entry point that:

* Parses URL parameters.
* Determines content to display.
* Routes to appropriate loader function.
* Handles direct item access.
* Content Loading.
* Content loaders
  + function loadCategoryContent(categoryId, showItemId)
  + function loadSubcategoryItems(categoryId, subcategoryId, showItemId)
  + function loadSubSubcategoryItems(categoryId, subcategoryId, subSubcategoryId, showItemId)

Hierarchical content loaders that:

* Fetch appropriate category/subcategory data.
* Render desktop and mobile views.
* Handle special items display.
* Support direct item highlighting.

3. UI RENDERING : Item Card Creation

* function createItemCard(item)

Generates HTML for both:

* + Desktop view (with quantity controls and cart button)
  + Mobile view (simplified for touch)
* function setupQuantitySelectors()

Initializes interactive quantity controls for items.

4. NAVIGATION : Basic Navigation

* function goBack() // Enhanced back handling:

// - Mobile/desktop differences

// - URL parameter cleanup

// - History management

* function goToMyTray() // Goes to cart page

Category Navigation

* function goToSubcategory(categoryId, subcategoryId)
* function goToSubSubcategory(categoryId, subcategoryId, subSubcategoryId)

Handles hierarchical menu navigation with proper URL parameter updates.

5. CAROUSEL FUNCTIONALITY

* function navigateCarousel(direction) // Moves carousel forward/back
* function goToCarouselItem(index) // Jumps to specific item
* function updateCarouselDisplay() // Updates carousel UI

Manages featured items carousel with smooth transitions.

6. MOBILE FUNCTIONALITY

* function showMobileItem(item) // Displays expanded mobile view with:

// - Image handling

// - URL state management

// - Cart synchronization

* function hideMobileItem() // Handles:

// - View cleanup

// - URL parameter removal

// - History management

Handles mobile-specific item display with:

* Image handling.
* Quantity adjustments.
* Cart operations.

7. CART INTERACTIONS

* function updateQuantity(itemId, change) // Adjusts item quantity
* function toggleCart(itemId) // Adds/removes item from cart

Provides direct cart manipulation from UI elements.

8. HELPER FUNCTIONS

Data Lookup

* function findCategory(categoryId)
* function findSubcategory(parentCategory, subcategoryId)
* function findItemById(itemId)
* function findItemInCategory(category, itemId)
* function findItemCategoryPath(itemId)

Navigates menu data hierarchy to locate specific items/categories.

Utilities

* function showError(message) // Displays error messages
* function loadFontAwesome() // Dynamically loads Font Awesome

9. EVENT LISTENERS

DOM Ready

* document.addEventListener("DOMContentLoaded")

Initializes:

* Back button handlers.
* Menu data verification.
* Mobile interaction handlers.
* Page content.
* Window Events.
* window.addEventListener("resize") // Handles mobile/desktop transitions
* window.addEventListener("cartUpdated") // Updates UI on cart changes
* window.addEventListener('popstate') // History navigation

10. GLOBAL EXPORTS

* window.navigateCarousel
* window.goToCarouselItem
* window.showMobileItem
* window.hideMobileItem

Makes key functions available for HTML event handlers.

Key Features:

* Responsive Design: Handles both desktop and mobile views.
* Hierarchical Navigation: Supports 3-level menu structure.
* Cart Persistence: Saves cart state across page refreshes.
* Dynamic Loading: Handles async menu data loading.
* Interactive Elements: Quantity selectors, carousel, mobile views.
* Error Handling: Graceful degradation for missing data.

6. mytray.js

Handles the "My Tray" page, where users review selections before checkout.

Key Functions:

* renderTrayItems() - Generates cart item displays.
* updateTrayQuantity() - Adjusts item quantities.
* removeFromTray() - Removes items from cart.
* updateTrayDisplay() - Refreshes the entire tray view.
* handleCheckout() - Initiates checkout process.

Key Features:

* Different layouts for desktop and mobile.
* Quantity adjustment.
* Total price calculation.
* Checkout preparation.

7. cart.js

Cart Object Methods:

* addItem() - Adds item to cart or increments quantity.
* removeItem() – Removes item from cart.
* updateQuantity() - Changes item quantity.
* getTotal() - Calculates cart total.
* getItemCount() - Returns total item count.
* saveToStorage() - Persists cart to localStorage.
* clearCart() - Empties the cart.

Key Features:

* localStorage persistence.
* Event-driven updates.
* Quantity management.
* Price calculations.

Event System

* Dispatches cartUpdated event whenever the cart changes.
* Listened to by category-page.js and mytray.js for UI updates.

8.menu-data.js

Data Structure & Organization

This file contains the entire menu structure in a hierarchical JSON format, organized into categories → subcategories → items.

Key Features:

* Global Accessibility: Attached to window.menuData for cross-file access.
* Event-Driven Loading: Dispatches menuDataLoaded event when ready.
* Special Items: Some items are marked as "specials" for featured display.

Structure Breakdown:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| id | string | Unique identifier for the category/item (e.g., "platters") |
| name | string | Display name (e.g., "Platters") |
| description | string | Optional item/category description |
| price | number | Item price (Ksh) |
| image | string | Path to the item image (e.g., "images/goatImage.png") |
| items | Array<Object> | List of items (if no subcategories) |
| subcategories | Array<Object> | Nested subcategories |

Contains all menu data in a structured format:

Data Structure:

* Categories (top-level menu sections).
* Subcategories (nested sections).
* Sub-subcategories (deeply nested sections).
* Items (individual menu items).

Key Features:

* Hierarchical organization.
* Comprehensive item details (name, price, description, image).
* Special items identification.
* Global accessibility via window object.

9. Reservation Process (reservation.html + reservation.js)

Key Features:

* Collects guest information (name, phone, date, time, party size)
* Offers payment options ("Pay Now" or "Pay on Arrival")
* Validates form inputs.
* Persists reservation data to localStorage.

Core Functions (reservation.js):

|  |  |
| --- | --- |
| Function | Description |
| processReservation() | Handles form submission and payment method selection |
| setupFormValidation() | Sets up real-time validation for form fields |
| goBack() | Navigation back to previous page |

Data Flow:

* User submits reservation form.
* Data is stored in localStorage under current-reservation.
* Redirects to either:
  + payment.html for "Pay Now"
  + subscription.html for "Pay on Arrival"

Mobile Considerations:

* Form fields stack vertically.
* Buttons hide on scroll down, reappear on scroll up
* Optimized touch targets for form inputs.

10.payment.html

Key Features:

* Payment method selection (M-Pesa or Credit Card).
* Dynamic form switching based on payment method.
* Input validation and formatting.
* Mock payment processing with 2-second delay.
* Cart clearing upon successful payment.

Payment Methods Implemented:

* M-Pesa:
* Phone number input with validation.
* Formatting for Kenyan phone numbers.
* Credit/Debit Card:
* Card number formatting (xxxx xxxx xxxx).
* CVV validation.
* Expiry date selection.
* Auto-filled amount from reservation.

Key Functions (embedded in payment.html):

* processPayment() //Handles payment processing for selected method
* goBack() //Navigation back to reservation page
* Dynamic form switching Toggles between M-Pesa and card forms

Data Flow:

* Retrieves reservation amount from current-reservation.
* Processes payment and stores transaction details.
* Clears cart via window.cart.clearCart().
* Redirects to subscription.html on success.

11. Subscription Confirmation (subscription.html + subscription.js)

Key Features:

* Personalized welcome message using reservation name.
* Subscription opt-in checkbox.
* Home navigation button.
* Reservation verification.

Core Functions (subscription.js):

* updateWelcomeMessage() //Personalizes greeting with guest name
* setupSubscriptionCheckbox() //Manages subscription opt-in state
* saveSubscriberInfo() //Stores subscriber data in localStorage
* verifyCartEmpty() //Ensures cart was properly cleared

Data Storage:

* Subscribers stored in wadau-subscribers array in localStorage.
* Includes name, contact info, and subscription date.

**12 .reservation.css**

Design Features:

* Responsive two-column layout (desktop) / stacked layout (mobile).
* Consistent styling across checkout flow.
* Decorative elements (swirl graphics, diamond image container).
* Adaptive button behavior (hide/show on scroll for mobile).

Key Style Components:

* Form input styling with bottom borders.
* Payment method selection dropdown.
* Pay Now/Pay Later button styles.
* Mobile-optimized layouts with larger tap targets.

**System-Wide Features.**

* Navigation & State Management.
* Consistent back/cancel buttons across all pages.
* Reservation data persists via localStorage.
* Cart automatically clears after successful payment.
* Deep linking support via URL parameters.

**Error Handling**

* Form validation with visual feedback.
* Try/catch blocks around localStorage operations.
* Fallback behaviors for missing data.

**Mobile Optimization**

* Responsive layouts for all screen sizes.
* Touch-friendly controls.
* Scroll-aware UI elements (hiding buttons on scroll down).
* Adaptive font sizes and spacing.