**1. Introduction**  
This documentation provides a comprehensive technical overview of the Food Library web application, designed to help users find meals by category or ingredient. The documentation covers all major aspects, including frontend code, APIs, analytics, and key functions with detailed explanations.

**2. Project Overview**  
Food Library is a React-free, Bootstrap-based web app with custom JavaScript that allows users to:  
• Search meals by category or ingredient  
• View search results with images and descriptions  
• Save favorite meals to an off-canvas menu  
• Access curated blog posts related to various food categories

**3. Technologies Used**  
• HTML5  
• CSS3 (with custom styles)  
• JavaScript (ES6)  
• Bootstrap 5.2 for UI components  
• Font Awesome for icons  
• Google Analytics for tracking  
• TheMealDB API for meal data

**4. Google Analytics Integration**  
Google Analytics is integrated using the Global Site Tag (gtag.js) for traffic and user behavior monitoring.

Html-

<!-- Google tag (gtag.js) -->

<script async src="https://www.googletagmanager.com/gtag/js?id=G-5GRS5L6BKV"></script>

<script>

window.dataLayer = window.dataLayer || [];

function gtag(){dataLayer.push(arguments);}

gtag('js', new Date());

gtag('config', 'G-5GRS5L6BKV');

</script>

This setup asynchronously loads the GA library and configures it with the tracking ID G-5GRS5L6BKV.

**5. API Details**  
Food Library uses TheMealDB API to fetch meal data.

**Base URL**  
https://www.themealdb.com/api/json/v1/1/

**Key Endpoints**  
• Filter by category: /filter.php?c={category}  
• Search by name or ingredient: /search.php?s={query}  
• Lookup meal details by ID: /lookup.php?i={id}

**API Response Example**

Json-

{

"meals": [

{

"idMeal": "52772",

"strMeal": "Teriyaki Chicken Casserole",

"strCategory": "Chicken",

"strInstructions": "Preheat oven to 350° F...",

"strMealThumb": "https://www.themealdb.com/images/media/meals/wvpsxx1468256321.jpg",

"strIngredient1": "soy sauce",

"strIngredient2": "water",

"strMeasure1": "3/4 cup",

"strMeasure2": "1/2 cup" ... } ]}

**6. JSON Structure Explanation**  
• idMeal: Unique meal ID  
• strMeal: Name of the meal  
• strCategory: Meal category  
• strInstructions: Cooking instructions  
• strMealThumb: Image URL  
• strIngredientN / strMeasureN: Ingredient and measurement pairs

**7. Key Functions with Code Snippets**

**performSearch()**  
Triggers when the search button is clicked. Determines search type and fetches meal data accordingly.

Javascript-

function performSearch() {

const query = document.getElementById('search-bar').value.trim();

const searchType = currentSearchType; // Either 'Category' or 'Meal Name'

if (!query) return;

let apiURL = '';

if (searchType === 'Category') {

apiURL = `https://www.themealdb.com/api/json/v1/1/filter.php?c=${encodeURIComponent(query)}`;

} else {

apiURL = `https://www.themealdb.com/api/json/v1/1/search.php?s=${encodeURIComponent(query)}`;

}

fetch(apiURL)

.then(response => response.json())

.then(data => showMealResults(data.meals))

.catch(err => console.error('API error:', err));

}

* Builds URL based on user input and search type.
* Fetches data from TheMealDB API.
* Passes meal data to showMealResults() for display.

**showMealResults(meals)**  
  
Displays meal cards dynamically on the page.

Javascript-

function showMealResults(meals) {

const mainDiv = document.getElementById('main');

mainDiv.innerHTML = ''; // Clear previous results

if (!meals) {

mainDiv.innerHTML = '<p>No meals found.</p>';

return;

}

meals.forEach(meal => {

const card = document.createElement('div');

card.className = 'card m-2';

card.style.width = '18rem';

card.innerHTML = `

<img src="${meal.strMealThumb}" class="card-img-top" alt="${meal.strMeal}">

<div class="card-body">

<h5 class="card-title">${meal.strMeal}</h5>

<button onclick="addToFavorites(${meal.idMeal})" class="btn btn-sm btn-warning">Add to Favorites</button>

</div>

`;

mainDiv.appendChild(card);

});}

* Clears old results before rendering new ones.
* Creates Bootstrap cards for each meal with image, title, and favorite button.

**addToFavorites(mealId)**  
Adds selected meals to a favorites list saved in localStorage.

Javascript-

function addToFavorites(mealId) {

let favorites = JSON.parse(localStorage.getItem('favorites')) || [];

if (!favorites.includes(mealId)) {

favorites.push(mealId);

localStorage.setItem('favorites', JSON.stringify(favorites));

alert('Added to favorites!');

} else {

alert('Meal already in favorites.');

}

}

**8. HTML Snippets**

**Search Section**

Html-

<div id="search-section" class="container text-center mt-4">

<div class="input-group">

<button class="btn btn-light dropdown-toggle" type="button" id="searchDropdown" data-bs-toggle="dropdown" aria-expanded="false">

Search By

</button>

<ul class="dropdown-menu" id="search-options">

<li><a class="dropdown-item" href="#" onclick="setSearchType('Category')">Category</a></li>

<li><a class="dropdown-item" href="#" onclick="setSearchType('Meal Name')">Meal Name/ Ingredient</a></li>

</ul>

<input type="text" id="search-bar" class="form-control" placeholder="Search for meals..." oninput="autocompleteSearch()">

<button class="btn btn-danger" type="button" onclick="performSearch()">

<i class="fa fa-search"></i>

</button>

</div>

<div id="autocomplete-dropdown" class="autocomplete-dropdown mt-2"></div>

</div>

This HTML defines a **Bootstrap Offcanvas** component that appears from the right side of the screen to display the user's **favorite meals**.

* **offcanvas offcanvas-end**: Initializes the offcanvas and positions it to slide in from the right.
* **tabindex="-1"**: Makes the component focusable for accessibility.
* **id="offcanvasNavbar"**: Unique identifier used to toggle or reference this panel.
* **aria-labelledby="offcanvasNavbarLabel"**: Links the panel to its heading for screen readers.

Inside the offcanvas:

* **Header (offcanvas-header bg-danger)**:
  + Displays the title “Favourites” with a red background (bg-danger) and white text.
  + Includes a close (btn-close) button that hides the panel when clicked.
* **Body (offcanvas-body)**:
  + Uses Flexbox (d-flex justify-content-center flex-wrap) to center and wrap its contents.
  + Initially shows a green loading spinner (spinner-border text-success) to indicate that favorite meals are being fetched.

**Favorites Offcanvas**

Html-

<div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasNavbar" aria-labelledby="offcanvasNavbarLabel">

<div class="offcanvas-header bg-danger">

<h5 class="offcanvas-title" id="offcanvasNavbarLabel" style="color:white;">Favourites</h5>

<button type="button" class="btn-close" data-bs-dismiss="offcanvas" aria-label="Close"></button>

</div>

<div id="favourites-body" class="offcanvas-body d-flex justify-content-center flex-wrap">

<div class="spinner-border text-success" role="status">

<span class="sr-only">Loading...</span>

</div>

</div>

</div>

**9. CSS Snippets**

Css-

#search-section {

max-width: 600px;

margin: auto;

}

.autocomplete-dropdown {

position: absolute;

background-color: white;

border: 1px solid #ddd;

max-height: 200px;

overflow-y: auto;

width: 100%;

z-index: 1000;

}

.card {

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

transition: transform 0.2s ease;

}

.card:hover {

transform: scale(1.05);

}

.offcanvas-header.bg-danger {

background-color: #dc3545 !important;

}

**#search-section** centers the search bar and restricts its width to a maximum of 600px for a balanced layout.

**.autocomplete-dropdown** is absolutely positioned beneath the search bar, styled with a white background and border, and limited to 200px height with vertical scrolling if needed. A high z-index ensures it appears above other elements.

**.card** applies a soft shadow and a smooth scale transition for visual depth and interactivity.

**.card:hover** slightly enlarges the card on hover to enhance user feedback and make the UI feel more dynamic.

**.offcanvas-header.bg-danger** forces the offcanvas header background to a specific Bootstrap red color (#dc3545) using !important to override any conflicting styles.

**10. Project Structure**

FoodLibrary

|- index.html

|- mealApp.css

|- Mealapp.js

|- logo.png.webp

|- favicon.ico

|- README.md

**11. Conclusion**  
This documentation summarizes the key technical details of Food Library, including frontend implementation, API integration, Google Analytics setup, and important code functions. It is intended to provide a clear understanding for developers and stakeholders reviewing the project.