Technical Design Document

Project: Varsity’s Eatery

Developer: Kayevon Azuca
Github: https://github.com/KayevonAzuca/VarsitysEatery

Introduction

What is the project?

It is a brochure website for the end-users but the site can also interact with an API to obtain relevant data and it provides a contact form that can be submitted and saved into a database which can be retrieved and displayed back to the end-user.

The project revolves around a business called Varsity’s Eatery which is a fast-food place that wants to display their eatery’s information such as their open hours, location images, history, etc. This is a place that wants to validate themselves online and provide potential customers knowledge about their products.

What is the project’s goals?

1) Provide informative services such as showing business and social information, images of the store’s eatery, and history on the business itself.

2) Display food items by their category. Include information such as a name, an image, description, and price per food item.

3) Provide a point of contact (or survey) for the end-users to submit their opinions on the eatery.

How was Varsity's Eatery built?

The project uses vanilla web technologies such as HTML, CSS, JavaScript, and PHP. More complex features involve fetching and receiving data from JSON files or a mySQL database. For example, food categories/items are fetched and cached so they could be dynamically displayed to the end-user. Contact forms are carefully validated both in the front-end and the back-end before being stored in the database.

Architecture

The following will go over the website’s features in detail by explaining the inner-workings/workflow of how web services are achieved.

Brochure Website

HTML & PHP

The front-end “.html” files are “.php” files. This is done to easily develop and create web pages by separating frequently used portions of the page into header, sidebar, and footer include files.

The header file contains the “<head></head>” tag that has values that are dynamically inserted depending on what page it is included with. Each web page contains PHP values that distinguish it from one another. This helps fill in the values that the header file needs.

For example the “<title>” tag value is filled in, the navigation highlight of the currently viewing page is active, and some page specific scripts are loaded.

CSS

The site is built around a mobile-first concept, sizing the website down to around 320px is possible. It also utilizes the use of CSS variables to help dynamically update HTML colors within the contact form to inform the user of any mistakes made when submitting a form.

JavaScript

The only script loaded in most web pages is “main.js”. This script contains multiple “Immediately-Invoked Function Expression” (IIFE) that each “lazy-load” their functionalities or “modules”. The modules are separated JavaScript files that are denoted as “ModuleName.mod.js”.

These modules have their own self contained functionalities that provide publicly exported functions and privately held variables to the IIFE. Modules used in this manner help keep data from being known in the global scope, and keeping them contained in an IIFE is another attempt at isolating the instruction used to provide features for the end-user.

Food Menu API

When an end-user loads the “Menu” page a few things will happen. An “Immediately-Invoked Function Expression” (IIFE) in “main.js” will detect an element with the ID of “menu”. This will allow further execution of the IIFE to obtain the “menu.json” file that is located in the back-end. This file contains all food categories and food items along with all relevant information to be display in the page.

The JavaScript module “Menu.mod.js” will be used to interact with PHP to obtain the JSON data. Initially, “SessionStorage.mod.js” will be used to check the browser’s session storage for the data. If an end-user visits the “menu.php” page once, JavaScript will fetch the JSON data and cache it in session storage. If the end-user continues to visit the “menu.php” page all on the same session then the data located in the session storage will always be used until the website tab is closed. This optimizes back-end usage by preventing JavaScript from fetching the same JSON data each time the end-user revisits the same page.

Using the JSON data and the “Menu.mod.js”, the menu can be dynamically created along with the categories navigation. The last thing that is checked is the URL parameter for “cat”. If the end-user in “index.php” clicked a category link that redirected them to “menu.php” then this “cat” value would be set to a category name that they would like to view on page load. This value is then extracted, validated, and used to display the default category.

Contact Form

If “main.js” finds contact form ID elements it will start to initialize and prepare for a contact form being submitted. The “Menu.mod.js” and “SessionStorage.mod.js” are loaded and used briefly to find the food categories to contain them in an array. They are then used to display checkbox inputs dynamically otherwise the contact form would be “unsumbittable”.

“SendContactForm.mod.js” handles contact form submissions and is only loaded/initialized when the end-user clicks on the form “submit” button. The module then caches text and input elements to later use to display any messages such as errors, warnings, or successions along with their corresponding color.

Form submissions are first validated in the front-end “SendContactForm.mod.js” module. This is done to lower the amount of back-end form validations until a guaranteed valid form is submitted. This valid form is still validated by PHP but most likely it will pass and be uploaded to the database.

The back-end uses the MVC pattern to upload the contact form data by interacting with the “Controller”. Retrieving a form to be displayed back to the end-user works in much the same way as sending a form. It uses “GetContactForm.mod.js” except it will only use the “Viewer” in the MVC pattern to obtain and format the response forms coming back as a response to JavaScript.