



FACULTY OF COMPUTING AND INFORMATICS

CIT4224

Internet and Web Publishing

Trimester 1, Session 2025 / 2026

Project Title: Coffee's Life

BY

Group: 5





No	Name	Student ID
1	Aziel Tan Zheng Chuan	241DC240LY
2	See Wing Kit	241DC240PJ
3	Vincent Lock Chun Kit	241DC2406W
4	Daniel Go Zi Yang	241DC240TF

Table of Contants

Table of Contants	2
TASK DISTRIBUTION FORM	4
PROJECT EVALUATION FORM	5
1. Executive Summary	7
2. Methodology	7
3. Screenshot and Layout.....	8
4. Detailed File Analysis	11
4.1. PHP Files Analysis (25 Files)	11
4.1.1. Core System & Utility Files (5 Files)	11
4.1.2. User Account Management (4 Files)	14
4.1.3. Products & Menu Display (3 Files)	16
4.1.4. Shopping Cart & Checkout Process (7 Files)	18
4.1.5. Reviews & Feedback System (4 Files)	22
4.1.6. Static Informational Pages (2 Files).....	25
4.2. CSS Files Analysis (10 Files)	26
4.2.1. Global Styles: Core Visual Foundation (1 File).....	26
4.2.2. Page-Specific Styles: Tailored Visuals (9 Files)	27
5. Identified Admin-Related Files.....	29
6. Administrator Role Overview	30
7. Admin Module Functionality Breakdown	31
8. Security Considerations	35
9. Database Interactions	36
10. Detailed Analysis of admin.css	37
10.1 General Structure and Theming	37

10.2	Key Component Analysis	37
10.3	Colour Palette and Typography.....	39
10.4	Responsive Design Approach	40
11.	Detailed Analysis of reviews.css.....	41
11.1.	General Structure and Theming	41
11.2.	Key Component Analysis.....	41
11.3.	Color Palette and Typography	43
11.4.	Responsive Design Approach	43
12.	Summary of Design Principles	44
13.	Architectural Observations & Best Practices (Combined)	45
14.	Conclusion	47
	APPENDIX.....	48

TASK DISTRIBUTION FORM

No.	Student ID	Student Name	Tasks Completed	Signature
1.	241DC240LY	Aziel Tan Zheng Chuan	ie. Homepage, JavaScript, drop-down menu Report, CSS Javascript php Database	
2.	241DC240PJ	See Wing Kit	Report, CSS, JavaScript, php Database	
3.	241DC2406W	Vincent Lock Chun Kit	Report, Homepage HTML, &CSS, Manu Homepage HTML, &CSS, php Database	
4.	241DC240TF	Daniel Go Zi Yang	Report, CSS, HTML	

CIT4224 Internet & Web Publishing

PROJECT EVALUATION FORM

STUDENT ID	STUDENT NAME	Peer Evaluation Total Marks (20M) (For Evaluator Use)	Project (For Evaluator Use) (40M)
241DC240LY	Aziel Tan Zheng Chuan		
241DC240PJ	See Wing Kit		
241DC2406W	Vincent Lock Chun Kit		
241DC240TF	Daniel Go Zi Yang		

1	Very Poor	2	Poor	3	Average	4	Good	5	Excellent
---	-----------	---	------	---	---------	---	------	---	-----------

WEBSITE:

Criteria	Marks	Comments/ Remarks
1. Content (User/Customer) <ul style="list-style-type: none"> Content relevance, appropriateness, organization and presentation 		
2. Content (Administrator): <ul style="list-style-type: none"> Content relevance, appropriateness, organization and presentation 		
3. Graphics & CSS		
4. Hyperlinks		
5. Form		
6. Multimedia elements		
7. Extra features (responsive element, javascript)		
8. Overall design		
Total (40)		

REPORT:

Criteria	Marks	Comments/ Remarks
• Checklist & Evaluation Form	/2	
• Task distribution	/2	
• Objective/Purpose	/2	
• Target User	/2	
• Screenshot of website	/10	
1. Cover page completed	/1	
2. Header, Page number and title numbering are included	/1	
Total (20)		

PRESENTATION:

Criteria	Marks	Comments/ Remarks
• Clarity of speaking	/3	
• Understanding of the website	/4	
• Presenter's appearance (formality, professionalism)	/3	
Total (10)		

1. Executive Summary

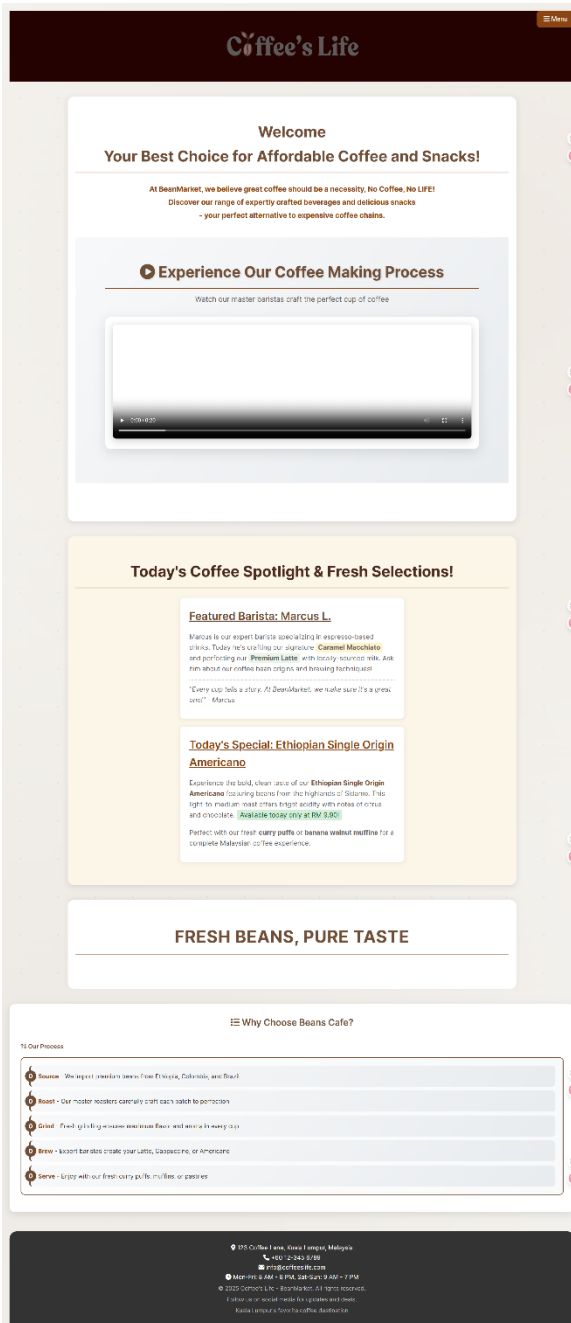
This report presents a holistic technical analysis of the "Coffee's Life" web application, an e-commerce platform developed for a coffee shop. The application's architecture leverages PHP for robust server-side processing and MySQL for data persistence, while its visual presentation is meticulously crafted using CSS. Key functionalities encompass secure user authentication, intuitive product browsing, a feature-rich shopping cart system with customization, and an interactive customer review platform. The codebase exhibits strong adherence to modern web development practices, including modular design, secure data handling (e.g., prepared statements for SQL, password hashing), comprehensive session management, and responsive frontend design with engaging aesthetics. This report details the purpose, functionalities, interactions, and best practices observed across all 25 PHP and 10 CSS files.

2. Methodology

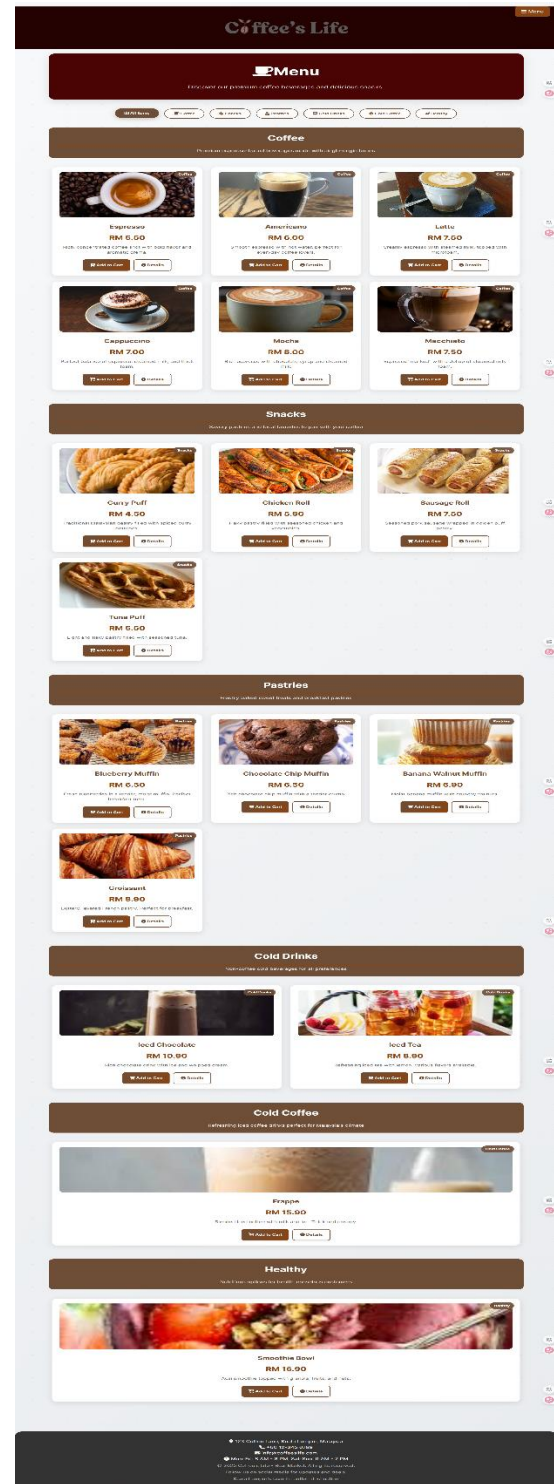
The analysis was conducted through a thorough review of all 35 provided source code files (25 PHP and 10 CSS). Each file was individually examined to ascertain its specific role, logic, data interactions, and dependencies. The files were then systematically categorized into logical groups based on their functional domain (e.g., Core System, User Accounts, Products, Cart, Reviews) and technology (PHP, CSS). This structured approach facilitates a clear and comprehensive understanding of the application's overall architecture, backend operations, and frontend presentation layers.

3. Screenshot and Layout

Home Page



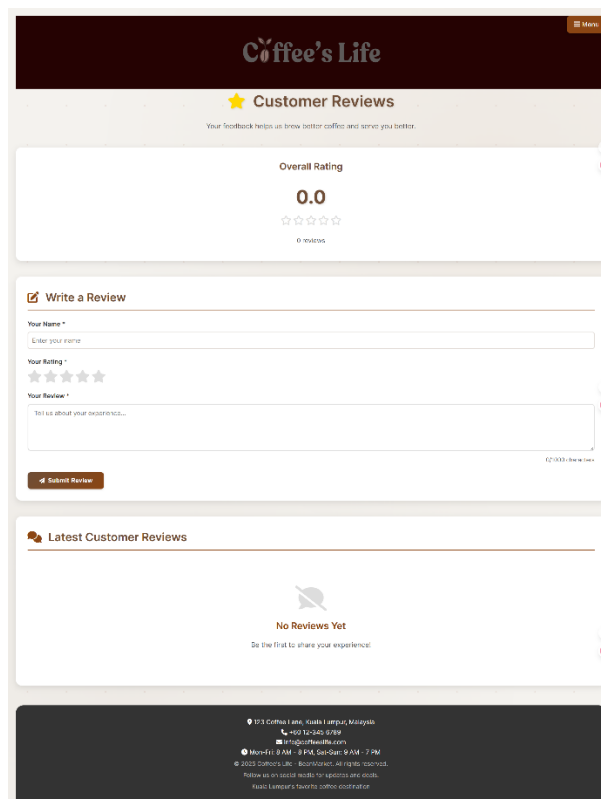
Menu Page



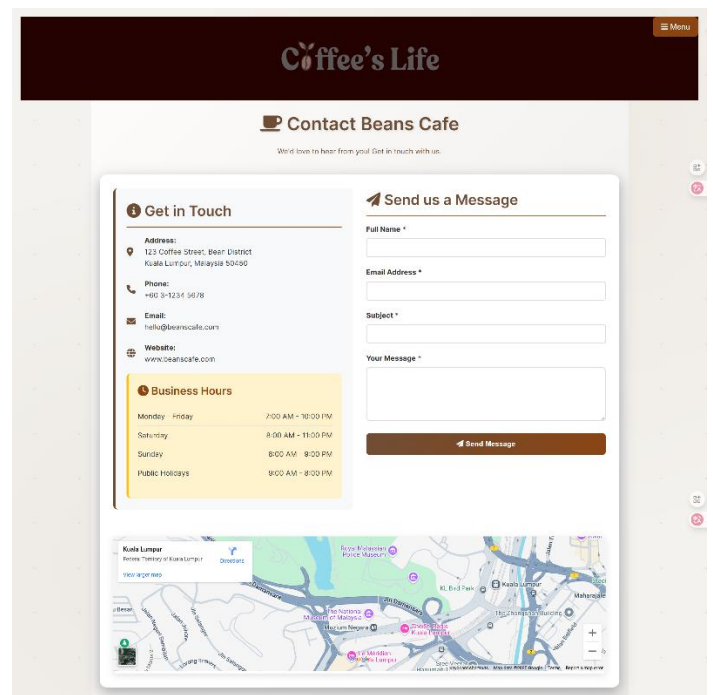
About Us Page



Review Page



Map & Location View Function



User Login Page

Add Cart Function Page

View Shopping Cart & Check Out Page

User Edit / Change Passwords

4. Detailed File Analysis

The 35 files are meticulously categorized and analysed below:

4.1. PHP Files Analysis (25 Files)

The PHP files form the server-side backbone, handling all business logic, database interactions, and dynamic content generation.

4.1.1. Core System & Utility Files (5 Files)

These files establish the fundamental infrastructure for the application, handling database connections and providing universally accessible, reusable functions.

- **config.php**
 - **Purpose:** Centralizes database connection parameters and establishes connections to the MySQL database.
 - **Details:** Defines connection credentials (\$host, \$dbname, \$username, \$password). Establishes two distinct database connections: a modern and secure **PDO (PHP Data Objects)** connection (configured for exceptions and associative fetches) used primarily by the reviews system, and a **MySQLi** connection for broader compatibility. Includes `error_log()` for connection failure diagnostics, differentiating messages for development vs. production environments.
 - **Interactions:** Essential for `functions.php`, `reviews_functions.php`, and `user_profile.php` to connect to the database.
- **functions.php**
 - **Purpose:** A comprehensive library of reusable PHP functions that encapsulate common operations across the entire website, promoting code reusability and maintainability.
 - **Details:** `require_once 'config.php';` to access database connections. Contains functions for:
 - **Product Management:** `getProducts()` (fetches all products via MySQLi), `getProductById($id)` (retrieves specific product details using MySQLi prepared

statements to prevent SQL injection), addProduct(...) (inserts new products via prepared statements).

- **Cart Management (Session-based):** Functions primarily manipulate the \$_SESSION['cart'] array. getCart() retrieves the cart, initializing it if necessary. addToCartEnhanced(\$item) adds or updates items with customization details. removeFromCart(\$productId), updateCartItemQuantity(\$productId, \$newQuantity), and clearCart() manage individual items or the entire cart.
- **Cart Calculations:** getTotalCartItems() sums quantities, and getTotalCartPrice() calculates the total monetary value by querying product prices from the database.
- **User Authentication/Registration:** registerUser(...) inserts new user data, securely hashing passwords with password_hash(). loginUser(\$username, \$password) authenticates users using password_verify() and sets \$_SESSION['user_id'] and \$_SESSION['user_logged_in']. checkUserExists(\$field, \$value) verifies if a username or email is already taken.
- **Interactions:** A core utility, accessed by almost all other PHP files that involve data fetching, cart operations, or user authentication.

- **header.php**

- **Purpose:** Provides a consistent navigation menu and branding elements that appear at the top of every webpage.
- **Details:** Initiates session_start() if a session is not already active. require_once 'functions.php'; to access utilities for dynamic content (e.g., cart item count, user login status). Dynamically sets the HTML <title> based on the current script's filename. Renders the main navigation (<nav>) with active link highlighting. Implements conditional logic to display "Dashboard" and "Logout" links for logged-in users, or "Login / Register" options for guests, and also shows the real-time count of items in the shopping cart.
- **Interactions:** Included at the beginning of nearly all public-facing PHP files.

- **footer.php**

- **Purpose:** Provides a consistent informational section displayed at the bottom of every webpage.
- **Details:** Contains static HTML for contact information (address, phone, email, operating hours), copyright details, and social media links (using Font Awesome icons). This standardizes the site's lower section.
- **Interactions:** Included at the end of many public-facing PHP files.

- **setup_database.php**

- **Purpose:** A utility script for initializing the entire database schema and populating it with initial data. This script is intended for one-time use during application setup.
- **Details:** Connects to the MySQL server without specifying a database, allowing it to create the beans_cafe database. It then reads and executes SQL commands from an external beans_cafe.sql file. This process includes creating all necessary tables (e.g., users, products, reviews, review_helpful, admins) and inserting sample data, including a default administrative account. The script incorporates try...catch blocks for robust error handling during database operations and file reading, providing detailed feedback on setup success or failure.
- **Interactions:** Standalone script, not part of the application's live operational flow.

4.1.2. User Account Management (4 Files)

These files handle user registration, login, logout, and profile viewing/editing, with a strong focus on secure handling of user credentials and personal data.

- **register.php**

- **Purpose:** Allows new users to create an account within the "Coffee's Life" system.
- **Details:** `require_once 'functions.php'`; to use user registration and existence check functions. Processes `$_POST` requests, collecting username, email, `contact_number`, password (and `confirm_password`), and address. Performs rigorous **server-side validation**, checking for empty fields, valid email format (`filter_var()`), uniqueness of username and email (`checkUserExists()`), and password matching. If all validations pass, it securely hashes the password using `password_hash()` and calls `registerUser()` to insert the new user data into the database. Upon successful registration, the user is redirected to `user_login.php`.
- **Interactions:** Communicates with `functions.php` for database operations and validation, and redirects to `user_login.php`.

- **user_login.php**

- **Purpose:** Authenticates existing users, granting them access to their accounts.
- **Details:** `session_start()`; is called at the very beginning to manage user sessions. `require_once 'functions.php'`; for the `loginUser()` function. It handles `$_POST` requests, retrieving username and password. Performs server-side validation for empty input fields. It then calls `loginUser()` to verify the provided credentials against the hashed passwords stored in the database. On successful login, it sets `$_SESSION['user_id']` and `$_SESSION['user_logged_in'] = true`; to maintain the user's logged-in state across pages. Users are then redirected to `user_profile.php` by default, or to a specific redirect URL if passed in the query string (e.g., from `checkout_prompt.php`). The page also provides convenient links for new users to `register.php` and for quick orders via `guest_checkout.php`.
- **Interactions:** Interacts with `functions.php` for authentication and session management. Redirects to `user_profile.php`, `register.php`, or `guest_checkout.php`.

- **user_logout.php**
 - **Purpose:** Facilitates the secure termination of a user's active session.
 - **Details:** `session_start()`; is invoked to access session data. The script then effectively logs out the user by unsetting all session variables (`$_SESSION = array();`) and destroying the session on the server (`session_destroy()`). After invalidating the session, the user is redirected to the `index.php` (homepage).
 - **Interactions:** Clears session data; redirects to `index.php`.

- **user_profile.php**
 - **Purpose:** Provides a personalized dashboard for logged-in users, enabling them to view and update their profile information and manage their password.
 - **Details:** `require_once` statements for `functions.php` and `config.php`. Implements robust access control, redirecting to `user_login.php` if the user is not authenticated (`!isset($_SESSION['user_logged_in']) || $_SESSION['user_logged_in'] !== true`). Fetches the current user's details from the database using their `$_SESSION['user_id']`. It handles `$_POST` requests for two distinct actions:
 - **Profile Updates:** Allows users to modify their email, `contact_number`, and address. It uses PDO prepared statements for secure `UPDATE` queries.
 - **Password Changes:** Requires users to input their current password for verification, then checks if the `new_password` matches `confirm_password` before hashing the new password and updating it securely in the database.
 - Success and error messages are managed via session variables and displayed to the user. Client-side JavaScript is included to handle tab switching (Profile, Orders, Password sections) and real-time validation for password confirmation.
 - **Interactions:** Retrieves user data from the database using `config.php`, updates data via PDO statements, and uses `$_SESSION` for user identification and message passing.

4.1.3. Products & Menu Display (3 Files)

These files are responsible for presenting the coffee shop's product catalog and individual product details to customers.

- **index.php**
 - **Purpose:** Serves as the main entry point and landing page of the "Coffee's Life" website, offering an inviting overview.
 - **Details:** Integrates header.php and footer.php to maintain consistent branding and navigation. It displays static marketing content designed to welcome visitors, including prominent announcement sections, promotional banners (often using placeholder images), and an embedded video showcasing the coffee-making process (<video> tag with a local MP4 source). The page also highlights key reasons to choose "Coffee's Life" through styled ordered and unordered lists, enhancing the user's understanding of the brand's values and processes.
 - **Interactions:** Primarily displays static and dynamic content, linking to other core pages.

- **menu.php**
 - **Purpose:** Displays the full list of products available for purchase, providing options for product customization and adding to the cart.
 - **Details:** Includes header.php. The current snippet shows products from a hardcoded \$products array, though in a production environment, this would ideally be populated dynamically by calling getProducts() from functions.php. It iterates through these products to display each item with its image, name, price, and description. A key feature is the JavaScript-controlled modal pop-up (addToCartModal) that appears when a user clicks "Add to Cart." This modal allows for detailed customization, including sugar_level, milk_type, special_instructions, and quantity selection. The form within this modal submits data to add_to_cart.php. Client-side JavaScript is extensively used for opening and closing the modal, dynamically updating the total price in the modal based on selected options, category filtering of products, and auto-hiding success/error messages that might be displayed from session data.
 - **Interactions:** Fetches product data (conceptually), uses JavaScript for dynamic UI,

and submits data to add_to_cart.php.

- **product_details.php**

- **Purpose:** Provides a comprehensive and detailed view of a single selected product.
- **Details:** require_once 'functions.php'; to access getProductById(). It identifies the specific product to display using an id passed as a \$_GET parameter in the URL. It calls getProductById() to fetch all available details for that product, including its description, ingredients, and preparation methods. If the product ID is invalid or the product is not found in the database, the script gracefully redirects the user back to menu.php. The page displays the product's image (with a placeholder.co fallback), name, price, and all extensive details. It also includes a dedicated form that submits to add_to_cart.php for adding the product directly to the cart from this page, with an option to select the quantity.
- **Interactions:** Retrieves product data via functions.php, displays it, and allows adding to cart via add_to_cart.php.

4.1.4. Shopping Cart & Checkout Process (7 Files)

These files manage the user's shopping cart, allowing modifications, and guiding them through the checkout process, including options for logged-in users and guests.

- **add_to_cart.php**

- **Purpose:** Processes requests from the client to add products to the shopping cart, including any customization options.
- **Details:** require_once 'functions.php'; (specifically for addToCartEnhanced()). session_start(); to manage the cart within the user's session. It handles \$_POST requests, retrieving the product_id, quantity, and detailed customization options such as sugar_level, milk_type, and special_instructions. It validates the product_id and quantity. A \$cartItem array is then constructed with all product and customization details. The addToCartEnhanced() function is called to integrate this item into the \$_SESSION['cart']. The script sets session-based success or error messages (\$_SESSION['cart_message']) based on the outcome. Finally, it redirects the user back to the originating page (\$_SERVER['HTTP_REFERER']) or to menu.php as a fallback.
- **Interactions:** Receives \$_POST data, updates \$_SESSION['cart'] via functions.php, sets session messages, and redirects.

- **remove_from_cart.php**

- **Purpose:** Handles the removal of a specific item from the user's shopping cart.
- **Details:** require_once 'functions.php'; (for removeFromCart()). session_start(); for session management. It processes \$_POST requests, extracting the product_id of the item to be removed. It then calls removeFromCart() to update the \$_SESSION['cart'] by removing the specified product. After the operation, the user is redirected back to view_cart.php.
- **Interactions:** Receives \$_POST data, updates \$_SESSION['cart'] via functions.php, and redirects.

- **update_cart.php**

- **Purpose:** Modifies the quantity of an existing item in the shopping cart.
- **Details:** `require_once 'functions.php';` (for `updateCartItemQuantity()`). `session_start();` to access the cart session. It handles `$_POST` requests, retrieving the `product_id` and `newQuantity` for the item. The `updateCartItemQuantity()` function is called to adjust the quantity of the item in `$_SESSION['cart']`; if `newQuantity` is 0, the item is effectively removed. The user is then redirected to `view_cart.php`.
- **Interactions:** Receives `$_POST` data, updates `$_SESSION['cart']` via `functions.php`, and redirects.

- **clear_cart.php**

- **Purpose:** Empties all contents from the user's shopping cart.
- **Details:** `require_once 'functions.php';` (for `clearCart()`). `session_start();` is called. It processes `$_POST` requests. The `clearCart()` function is invoked, which resets the `$_SESSION['cart']` array to an empty state. After the cart is cleared, the user is redirected back to `view_cart.php`.
- **Interactions:** Receives `$_POST` data, resets `$_SESSION['cart']` via `functions.php`, and redirects.

- **view_cart.php**

- **Purpose:** Displays the current contents of the user's shopping cart, allowing for review and progression to checkout.
- **Details:** `require_once 'functions.php';`. `session_start();`. It implements crucial access control: if the user is neither logged in (`!isset($_SESSION['user_logged_in'])`) nor explicitly marked as a guest (`!isset($_SESSION['is_guest'])`), they are redirected to `checkout_prompt.php` to initiate the checkout flow. It retrieves the current cart contents using `getCart()` and calculates the `totalPrice` using `getTotalCartPrice()`. The page then iterates through the `$_SESSION['cart']` to display each item, its quantity, customizations, and individual price. For each item, forms are provided to allow users to update quantities (submitting to `update_cart.php`) or remove items (submitting to `remove_from_cart.php`). A clear overall cart summary and total price are displayed. An instruction regarding "Pay at Counter" suggests the intended payment model.

Action buttons are provided for "Continue Shopping" (linking to menu.php), "Clear Cart" (submitting to clear_cart.php), and "Proceed to Checkout" (linking to checkout_prompt.php).

- **Interactions:** Retrieves cart data via functions.php, displays dynamic content, and directs to other cart/checkout related pages.

- **checkout_prompt.php**

- **Purpose:** Guides the user on how to proceed with the checkout process by offering options to log in, register, or continue as a guest.
- **Details:** session_start();. It includes an initial redirection mechanism: if the user is already logged in (\$_SESSION['user_logged_in'] === true) or has already indicated they are a guest (\$_SESSION['is_guest'] === true), they are immediately redirected to view_cart.php (or a specified redirect URL), bypassing this prompt. Otherwise, the page presents clear options as hyperlinks: "Login" (linking to user_login.php), "Create Account" (linking to register.php), and "Guest Checkout" (linking to guest_checkout.php). Importantly, a redirect URL parameter is appended to these links, ensuring that after the user completes their authentication or guest information, they are returned to the view_cart.php to finalize their order.
- **Interactions:** Acts as a gateway to login, registration, or guest checkout, influencing user flow.

- **guest_checkout.php**

- **Purpose:** Collects minimal contact information (specifically a mobile phone number) from users who choose to proceed with their order as guests, without creating a full account.
- **Details:** session_start(); and require_once 'functions.php';. Similar to checkout_prompt.php, it contains redirection logic to send users to view_cart.php if they are already logged in or have previously been identified as a guest. It handles \$_POST requests, collecting the contact_number from the form. Server-side validation is performed on the contact_number (checking for emptiness and a basic phone number format using preg_match). If the validation passes, \$_SESSION['is_guest'] = true; is set, and the \$_SESSION['guest_contact_number'] is

stored. The user is then redirected to view_cart.php or a specific redirect URL.

- **Interactions:** Receives \$_POST data, validates it, sets guest-related session variables, and redirects.

4.1.5. Reviews & Feedback System (4 Files)

These files implement the functionality for customers to submit reviews and ratings, and for these reviews to be displayed and managed.

- **rating_comments.php**

- **Purpose:** Allows users to submit new reviews and provides a display for existing customer reviews and ratings.
- **Details:** require_once 'functions.php'; and require_once 'reviews_functions.php';. session_start();. It identifies the current user (by \$_SESSION['user_id'] if logged in, or by \$_SERVER['REMOTE_ADDR'] for guests). It handles \$_POST requests for review submissions, collecting rating, comment, and guest_name (if the user is not logged in). Server-side validation is performed on the input by calling validateReviewInput() from reviews_functions.php (checking rating range, comment length, and guest name presence). It also uses hasUserReviewed() to prevent users/IPs from submitting multiple reviews within a 24-hour period. If validation and spam checks pass, addReview() is called to save the feedback to the database. Success or error messages are managed and displayed to the user. The page includes a form for new review submissions and displays a list of existing reviews (which are likely fetched using getLatestReviews() or getProductReviews()). Client-side JavaScript is integrated for interactive star rating input, real-time form validation, and dynamic show/hide of the guest name field based on the user's login status.
- **Interactions:** Receives \$_POST data, validates it via reviews_functions.php, calls addReview(), and displays content.

- **reviews_functions.php**

- **Purpose:** Contains the core backend logic and utility functions specifically for managing review data within the database.
- **Details:** require_once 'config.php'; to access the \$pdo database connection. This file provides critical functions:
 - **addReview(...):** Inserts a new review record into the reviews table using a PDO prepared statement for secure data insertion. It supports both logged-in users (by user_id) and guest submissions (by guest_name and ip_address), setting the

review status to 'approved' by default.

- `getLatestReviews($limit)`: Fetches the most recent approved reviews from the database, performing a LEFT JOIN with the users table to display associated usernames.
- `getProductReviews($product_id, $limit, $offset)`: Retrieves reviews specific to a given product, with support for pagination (limit, offset) to efficiently load large sets of reviews.
- `getTotalReviewsCount($product_id)`: Returns the total count of approved reviews for a specific product or across all products.
- `getAverageRating($product_id)`: Calculates the average star rating for a product.
- `hasUserReviewed($user_id, $ip_address)`: Implements a spam prevention mechanism by checking if a user (by ID) or an IP address (for guests) has submitted a review within the last 24 hours.
- `validateReviewInput(...)`: Performs comprehensive server-side validation on review data, ensuring the rating is within range, the comment meets minimum/maximum length requirements, and a guest name is provided if necessary.
- `getUserIpAddress()`: A helper function to reliably retrieve the client's IP address.
- `getReviewDisplayName($review)`: Formats the name to be displayed for a review, prioritizing the username, then the guest name, and defaulting to "Anonymous" if neither is available.
- **Interactions:** Directly interacts with the database (\$pdo) and provides API for other PHP files to manage reviews.
- **load_more_reviews.php**
 - **Purpose:** An AJAX endpoint designed to dynamically load additional customer reviews without requiring a full page reload, thereby improving user experience and page performance.
 - **Details:** `require_once 'config.php';` and `require_once 'reviews_functions.php';`. It sets the Content-Type header to application/json to inform the client that the response will be in JSON format. It strictly processes only GET requests. It retrieves offset and limit parameters from the GET request to implement pagination, determining which batch

of reviews to fetch. It then calls `getProductReviews()` from `reviews_functions.php` to retrieve the next set of reviews. The script determines if there are more reviews available (`$has_more` flag) by comparing loaded count with total review count. Finally, it returns a JSON-encoded response containing the fetched reviews data, a success status, the `has_more` flag, `total_loaded` count, and `total_reviews` count. It also includes a `generateStarsHTML()` helper function to construct the HTML for star rating visuals.

- **Interactions:** Serves as a backend API endpoint for client-side AJAX calls.
- **mark_helpful.php**
 - **Purpose:** An AJAX endpoint that allows users to mark a review as "helpful," contributing to the relevance and credibility of feedback.
 - **Details:** `require_once 'config.php';` and `require_once 'reviews_functions.php';`. It sets the Content-Type header to `application/json` and only accepts POST requests. It retrieves the `review_id` from the JSON-encoded request body. To prevent duplicate votes, it identifies the user by `$_SESSION['user_id']` (if logged in) or by `session_id()` (for guests). A query is performed on the `review_helpful` table to check if the specific user or session has already marked this review as helpful. If not already marked, a new record is inserted into the `review_helpful` table. The `helpful_count` for the corresponding review in the `reviews` table is then updated (implicitly or by fetching the updated count after the insert). Finally, it returns a JSON-encoded response indicating success, a confirmation message, and the `new_count` of helpful votes.
 - **Interactions:** Serves as a backend API endpoint for client-side AJAX calls to record helpful votes.

4.1.6. Static Informational Pages (2 Files)

These files primarily provide content-focused information about the business, with minimal dynamic server-side processing.

- **about_us.php**
 - **Purpose:** Presents information about "Coffee's Life - Coffee's Life" to visitors.
 - **Details:** Includes header.php and footer.php. Contains static HTML content detailing the company's mission, values (sustainability, quality), and vision. Includes direct links to social media profiles.
 - **Interactions:** Displays static content.
- **contact_us.php**
 - **Purpose:** Allows users to send messages or inquiries to the coffee shop.
 - **Details:** require_once 'functions.php'; session_start();. Handles \$_POST requests: collects name, email, subject, and message. Performs **server-side validation** (empty checks, email format using filter_var()). (The current snippet focuses on validation and form display; typically, the next step would be sending an email or storing the message.) Displays success (\$success_msg) or error messages (\$name_err, etc.). Client-side JavaScript is integrated for real-time form validation (checking name length, email format, message length) before submission, providing immediate feedback to the user, and includes simple CSS animations.
 - **Interactions:** Processes form submissions, validates input, and potentially integrates with an email sending mechanism.

4.2. CSS Files Analysis (10 Files)

The CSS files are responsible for the aesthetic presentation, layout, and user experience of the "Coffee's Life" web application. They define the visual theme, responsiveness, and interactive elements.

4.2.1. Global Styles: Core Visual Foundation (1 File)

This stylesheet establishes the foundational visual identity and common styling patterns applied across the entire website.

- **global.css**
 - **Purpose:** The primary stylesheet, defining universal CSS variables, baseline typography, common layout patterns, and reusable component styles for the entire "Coffee's Life - Coffee's Life" website.
 - **Key Features & Logic:**
 - **CSS Variables (:root):** Defines a comprehensive set of custom properties (variables) for the coffee-themed color palette (--primary-brown, --secondary-brown, etc.), consistent spacing (--spacing-xs to --spacing-xl), border radii (--border-radius-sm to --border-radius-lg), and box shadows (--shadow-light to --shadow-heavy). This enables highly efficient and consistent theming.
 - **Base Styles:** Universal box-sizing, default font-family ('Inter'), margins, padding, background-color (--cream-bg), and line-height. A subtle, fixed coffee bean SVG pattern is applied via a body::before pseudo-element for background texture.
 - **Heading & Link Styles:** Defines consistent styles for all h1-h6 elements (color, weight, margins) and a (link) elements (color, text-decoration, transition effects). h1 includes a text-shadow.
 - **Reusable Components:** Provides styling for common UI elements: containers, sections, cards, product grids, tables, and forms.
 - **Themed Buttons:** Highly stylized buttons with linear gradients, shadows, and unique hover effects (e.g., "steam rise" using a ::before pseudo-element), and various semantic color variants (.primary, .success, .danger, .secondary).
 - **Messages & Alerts:** Consistent styling for .message-box and .alert (success, error, warning, info) with distinct background colors, border colors, and text

colors for clear user feedback.

- **Animations:** Defines keyframe animations (fadeIn, coffeeFloat, steamRise, beanBounce) and applies them to core elements like containers, product items, and buttons, adding dynamic visual interest.
- **Custom Scrollbar:** Styles the browser's scrollbar to match the coffee theme.
- **Responsive Design:** Foundational @media queries for overall layout and typography adjustments on different screen sizes (max-width: 768px, 480px).
- **Interactions:** Serves as the primary source of theme variables and base styles, implicitly influencing all other stylesheets.

4.2.2. Page-Specific Styles: Tailored Visuals (9 Files)

These stylesheets apply unique designs and overrides for specific pages or major sections of the website, building upon the global foundation.

- **about-us.css**
 - **Purpose:** Styles the "About Us" page content and layout.
 - **Features:** Specific linear gradient background override for the body, stylized content containers, and custom social media icon styling (including Instagram gradient effect) with hover animations.
 - **Interactions:** Overrides/extends global.css for body background and specific component styling.
- **cart.css**
 - **Purpose:** Styles the "View Cart" page, detailing the appearance of cart items, summaries, and actions.
 - **Features:** Prominent cart header, structured cart item display (CSS Grid), detailed item info, quantity controls, and a prominent total amount. Responsive adjustments for mobile.
 - **Interactions:** Utilizes CSS variables from global.css.
- **checkout.css**
 - **Purpose:** Styles the various pages involved in the checkout process, including the prompt, guest checkout form, and checkout summary.
 - **Features:** Themed containers, elegant "checkout option" cards with a "sliding" gradient hover effect, structured guest form, and a visual checkout progress indicator.

Extensive responsiveness across multiple breakpoints.

- **Interactions:** Heavily relies on CSS variables from global.css.
- **contact-us.css**
 - **Purpose:** Styles the "Contact Us" page, including the contact form, informational sections, and business hours.
 - **Features:** Sets a distinct background for the body and styles the main .contact-container as a CSS Grid for content layout. Includes stylized contact details with icons, themed form inputs, a gradient submit button, and a map container.
 - **Interactions:** Inherits from global.css for base styles and uses its variables.
- **dashboard.css**
 - **Purpose:** Styles the user's "Dashboard" page (user_profile.php), showing user stats, recent orders, and quick actions.
 - **Features:** Themed header with a subtle SVG background pattern, statistical cards with icons and hover effects, grid layouts for content sections, and internal navigation. Uses fadeIn and bounce animations for dynamic display.
 - **Interactions:** Uses variables from global.css for colors, spacing, and shadows.
- **header.css** (Note: Content largely subsumed by home_page.css and global.css for primary styling; acts as a placeholder or minor specific styles.)
 - **Purpose:** Primarily handles the layout and responsiveness of the main header component.
 - **Features:** Defines styles for the main header bar, logo, and navigation menu within the header.php file, ensuring its appearance and behavior.
 - **Interactions:** Works in conjunction with global.css and home_page.css.
- **home_page.css**
 - **Purpose:** Styles elements specific to the main index.php (Home Page), often overriding or extending global styles to create a unique first impression.
 - **Features:** Hero section with overlay, main logo styling, fixed menu toggle and off-canvas sidebar menu, image slider, detailed content sections (announcements, barista spotlight, promotions). **Crucially, includes unique custom list styles (ordered, unordered, image-based) with coffee-themed icons (emoji, rotating SVG) and animations.**
 - **Interactions:** Builds upon and sometimes overrides styles from global.css.

- **menu.css**
 - **Purpose:** Styles the product menu page, including categories, product cards, and the add-to-cart modal.
 - **Features:** Themed header, category filter buttons with active states, responsive product grid with rich hover effects (transform, shadow, border color), product details truncation, and comprehensive modal styling for add-to-cart customization.
 - **Interactions:** Relies heavily on variables from global.css.
- **product-details.css**
 - **Purpose:** Styles the dedicated page for individual product details.
 - **Features:** Two-column grid layout for product image and info, elegant ::before gradient border effect, clear product information sections, and stylized quantity selector with action buttons.
 - **Interactions:** Utilizes CSS variables from global.css.

5. Identified Admin-Related Files

Identification of Admin-Related Files

The following files within the project codebase are integral to or directly support the administrative functionalities of the "Coffee's Life" e-commerce website:

- **requirements.md:** This document defines the project scope, including the specifications and guidelines for the implementation of administrative features.
- **admin_dashboard.php:** Serves as the primary landing page for administrators, offering a concise overview of key operational metrics and providing direct access to essential administrative tasks.
- **admin_login.php:** Manages the authentication protocol for administrative users, ensuring secure access to the backend system.
- **admin_logout.php:** Facilitates the secure termination of an administrator's session.
- **admin_orders.php:** Dedicated to the management and display of customer order information.

- **admin_products.php**: Enables administrators to perform **CRUD (Create, Read, Update, Delete)** operations on products within the e-commerce catalog.
- **admin_users.php**: Provides tools for the oversight and management of registered user accounts.
- **functions.php**: Contains a repository of shared helper functions, which include functionalities critical for administrator authentication and data retrieval across the module.
- **config.php (Assumed)**: While not explicitly provided, this file is typically responsible for storing crucial configuration parameters, such as database connection details and potentially sensitive administrative credentials or settings, which are vital for the module's operation.

6. Administrator Role Overview

Based on the specifications outlined in **requirements.md**, the Administrator role is defined by the following core responsibilities and privileges:

- **System Management**: Possesses comprehensive control over the core data elements of the e-commerce system.
- **Product Management**:
 - Authority to **add new products** to the online catalog.
 - Ability to **modify existing product details**, including but not limited to name, price, description, and imagery.
 - Capacity to **remove products** from public availability.
- **Order Management**:
 - Capability to **view all customer orders** placed through the platform.
 - Privilege to **update order statuses** (e.g., from 'Pending' to 'Processing', 'Shipped', 'Completed', or 'Cancelled').

- Access to **detailed order information**, encompassing customer particulars and itemized purchases.
- **User Management:**
 - Access to **view all registered user accounts**.
 - The capacity to perform administrative actions on user accounts (e.g., managing roles, account blocking, or password resets), though specific actions may be restricted to viewing functionalities within the scope of the provided files.
- **Database Interaction:** Engages in direct interaction with the **MySQL database** to execute **CRUD** operations on product, order, and user data.
- **Security Adherence:** A strict emphasis is placed on prioritizing **security best practices** in all coding and operational procedures, particularly concerning authentication and data handling.

7. Admin Module Functionality Breakdown

A detailed examination of each primary administrative component reveals the following functionalities:

- **admin_login.php**
 - **Purpose:** To securely authenticate administrative users before granting access to the system's backend.
 - **Functionality:**
 - Presents a dedicated login interface for username and password submission.
 - Validates provided credentials against a predefined dataset (e.g., 'admin'/'admin' for demonstrative purposes).
 - Upon successful authentication, a secure PHP session is initiated for the administrator (`$_SESSION['admin_logged_in'] = true`).
 - Authenticated users are subsequently redirected to the `admin_dashboard.php` page.

- Handles and displays appropriate error messages for invalid credentials or incomplete input fields.
- **admin_dashboard.php**
 - **Purpose:** To serve as the central hub for administrative activities, offering an at-a-glance overview of critical system metrics.
 - **Functionality:**
 - Displays key performance indicators and statistics (e.g., total products, total orders, total users, cumulative sales).
 - Presents a concise list of recent orders for immediate review.
 - Provides intuitive navigation links to other administrative sections (Products, Orders, Users).
 - Offers a personalized greeting to the logged-in administrator.
- **admin_products.php**
 - **Purpose:** To facilitate the comprehensive management of product inventory and details available on the website.
 - **Functionality:**
 - Enables administrators to **add new products** by inputting essential details such as name, price, category, description, ingredients, preparation method, and an associated image.
 - Supports the **modification of details** for existing products.
 - Provides the capability to **delete products** from the database.
 - Manages **image uploads** for products, ensuring proper file storage.
 - Presents a tabular listing of all products, each with options for editing or deletion.
- **admin_orders.php**

- **Purpose:** To provide administrators with robust tools for monitoring and managing customer orders.
- **Functionality:**
 - Lists all recorded orders, typically including information such as order ID, customer name, total amount, current status, and creation date.
 - Allows administrators to access **detailed information** for specific orders, encompassing all purchased items, their quantities, and any customizations.
 - Offers functionality to **update the status of an order** (e.g., from 'Pending' to 'Processing' or 'Completed').
 - Displays relevant order statistics.
- **admin_users.php**
 - **Purpose:** To manage registered user accounts on the platform.
 - **Functionality:**
 - Displays a comprehensive list of all registered users.
 - Presents key user details such as username, email address, and registration date.
 - Provides an immediate overview of top users by order count and recent registrations.
 - *(Note: While potential enhancements could include direct editing of user profiles, blocking users, or resetting passwords, these are not explicitly implemented within the provided code snippets for direct modification.)*
- **admin_logout.php**
 - **Purpose:** To securely terminate an administrator's active session.
 - **Functionality:**
 - Clears all session variables pertinent to the administrator's login state.
 - Destroys the current session to prevent unauthorized access.

- Redirects the user to the admin_login.php page, ensuring no residual session data persists.
- **functions.php**
 - **Purpose:** To serve as a utility file, housing reusable functions that support various components of the application, including the administrative module.
 - **Functionality (Relevant to Admin Module):**
 - **checkAdminAuth():** A critical function designed to verify the login status of an administrator. It ensures that unauthorized access attempts to administrative pages are redirected to the login interface.
 - **getAdminStats():** Retrieves statistical data for presentation on the dashboard.
 - **getAllOrdersAdmin():** Fetches order data specifically for the administrative panel.
 - **addProductAdmin(), updateProductAdmin(), deleteProductAdmin():** Functions responsible for handling product management operations.
 - **handleImageUpload():** A helper function dedicated to managing the upload of product images.
 - **getAllUsersAdmin():** Retrieves user data for administrative oversight.
 - (Includes other database interaction functions that facilitate reading from and writing to data tables for products, orders, and users).

8. Security Considerations

The **requirements.md** document explicitly emphasizes the critical importance of "Security Reminder: Always prioritize security in coding." For the administrative module, robust security measures are paramount due to its direct access to sensitive data and critical system functionalities. Key security considerations implemented or implied by the provided files include:

- **Authentication:**
 - **Dedicated Admin Login (admin_login.php):** Establishes a distinct access point for administrators, segregating it from general user logins.
 - **Session Management:** Utilizes PHP sessions to maintain the integrity of the administrator's logged-in state.
 - **Session Regeneration (session_regenerate_id(true)):** Applied post-successful login to mitigate **session fixation attacks** by generating a new session ID.
 - **Credential Handling:** While a simple 'admin'/'admin' credential check is functional for a diploma assignment, a production-level application would necessitate a secure, database-backed user table for administrative accounts, employing strong **password hashing** techniques (e.g., password_hash()) to protect credentials.
- **Authorization:**
 - **checkAdminAuth():** A crucial function (located in functions.php) that rigorously restricts access to administrative pages, ensuring that only authenticated administrators can view or perform privileged actions.
- **Input Validation & Sanitization:**
 - **htmlspecialchars() and trim():** Employed consistently throughout the codebase (e.g., in admin_login.php, admin_products.php) to prevent **XSS (Cross-Site Scripting)** vulnerabilities and perform basic string cleaning. This is essential for all user-supplied data, including product descriptions, names, and various input fields.

- **Type Casting ((int)):** Applied to numeric inputs such as `product_id` and `quantity` to enforce data type integrity, thereby helping to prevent **SQL injection vulnerabilities**.
- **File Upload Security (`handleImageUpload()`):**
 - Incorporates checks for file upload errors.
 - *(For comprehensive security, this function should ideally include rigorous checks for allowed file types, size limitations, and the renaming of uploaded files to prevent the execution of malicious scripts and ensure consistent storage conventions).* The current implementation indicates basic handling of image uploads, which must be secured.

9. Database Interactions

The administrative module is characterized by its extensive reliance on interactions with the **MySQL database**. All core operations—including adding products, updating order statuses, and viewing user information—involve fundamental database interactions:

- **Reading Data:** Pertains to the retrieval of product listings, detailed order information, user account specifics, and statistical data for reporting.
- **Writing Data:** Encompasses the insertion of new product records, the modification of existing product details, and the alteration of order statuses.
- **Deleting Data:** Refers to the removal of product records from the database.

These interactions are systematically abstracted through dedicated functions within **functions.php**. These functions encapsulate the **SQL queries** and manage the database connection (established via `config.php`), promoting modularity, centralizing database logic, and enhancing maintainability.

10. Detailed Analysis of admin.css

This section provides an in-depth examination of the `admin.css` file, which governs the styling of the "Coffee's Life" administrative panel.

10.1 General Structure and Theming

The `admin.css` file establishes a **professional, clean, and warm aesthetic**, aligning with a coffee-themed brand identity. The strategic application of **browns, creams, and gold accents** creates a cohesive visual identity that reinforces the brand's character. The overall layout adopts a **desktop-first design approach**, supplemented by clearly defined responsive adjustments to ensure optimal display across various device types.

10.2 Key Component Analysis

A breakdown of the styling applied to key components within `admin.css` is as follows:

- **Body & Container:**
 - The `.admin-body` utilizes a subtle linear-gradient background (`#f4f2ee` to `#e8e3dd`), suggesting a light, textured surface.
 - `min-height: 100vh` is employed to guarantee that the background consistently covers the entire viewport.
 - The font-family declaration specifies 'Inter' as the primary typeface, a modern and highly readable sans-serif font suitable for administrative interfaces.
 - The `.admin-container` centers content and applies a max-width of 1200px, defining a structured content area.
- **Header:**
 - The `.admin-header` features a prominent brown linear-gradient (`#6f4e37` to `#8b4513`), serving as a strong brand element.
 - The `h1` element incorporates a gold icon (`#ffd700`), further reinforcing the "Coffee's Life" branding.
 - A soft box-shadow (`0 10px 30px rgba(0, 0, 0, 0.1)`) is applied to add depth to the header.

- **Navigation:**

- The .admin-nav is styled as a distinct white card, incorporating padding, border-radius, and box-shadow.
- Navigation links (.admin-nav a) are designed as visually appealing buttons with a brown linear-gradient, border-radius, and a subtle translateY hover effect accompanied by an increased box-shadow, providing effective visual feedback. An active state is implemented to clearly indicate the current page.

- **Cards & Stats:**

- .admin-card elements are visually distinct with white backgrounds, rounded corners, padding, and box-shadow, and are separated by margin-bottom.
- The .stats-grid effectively leverages CSS Grid for a flexible layout of .stat-card elements. These cards echo the header's brown gradient and display large, bold, gold numbers (.stat-number), effectively highlighting key metrics.

- **Forms:**

- The .admin-form is constrained to a 600px width for improved readability and user focus.
- Input elements (input, textarea, select) are consistently styled with ample padding, rounded border-radius, light borders, and a subtle background (#fafafa). Focus states are clearly defined with a brown border (#6f4e37) and a subtle box-shadow, providing essential usability cues.
- A custom .file-upload area is designed with a dashed border and hover effects to enhance the visual appeal of file selection.

- **Tables:**

- The .admin-table presents data with clarity, utilizing border-collapse, appropriate padding, and an alternating background on hover (#f8f6f3).
- Table headers are made prominent with a brown linear-gradient background and white text, maintaining brand consistency.
- Images within tables are styled to be small and rounded for a clean presentation.

- **Buttons:**

- A comprehensive set of .btn styles (primary, secondary, danger, success, small) ensures consistent interactive elements. All buttons feature linear-gradient

backgrounds, border-radius, and satisfying hover effects (translateY, box-shadow), clearly indicating interactivity.

- **Status Badges & Messages:**

- The .status-badge provides clear visual indicators for various states (active, inactive, pending, completed) through distinct background and text colors.
- .message classes (success, error, info) offer standardized feedback mechanisms to the user.

- **Login Form:**

- The .admin-login-container is a standalone, well-designed modal/card for login, reflecting the overall admin panel's aesthetic with generous padding, border-radius, box-shadow, and a prominent brown border.
-

10.3 Colour Palette and Typography

The color palette and typography choices for admin.css are as follows:

- **Primary Colors:** #6f4e37 (Primary Brown), #8b4513 (Secondary Brown), and #a0522d (Darker Brown) constitute the core of the brand's identity.
- **Accent Color:** #ffd700 (Gold) is strategically used for icons and highlighted numerical data, providing warmth and emphasizing importance.
- **Backgrounds:** #f4f2ee, #e8e3dd, #fafafa, and white are employed for light, clean sections.
- **Text Colors:** Primarily #333 and #666 are used to ensure optimal readability.
- **Typography:** 'Inter' is selected as the main font, recognized for its clean lines and legibility, making it highly suitable for administrative interfaces.

10.4 Responsive Design Approach

admin.css incorporates **media queries** for max-width: 768px and max-width: 480px to ensure comprehensive responsiveness. Key responsive adjustments include:

- Adjustments to padding.
- Scaling of header font sizes.
- Vertical stacking of navigation links (flex-direction: column) on smaller screens.
- The .stats-grid collapsing to a single column (1fr) for mobile viewing.
- Reduction of table font sizes and padding to accommodate smaller screens.

This approach guarantees that the admin interface remains highly usable and visually appealing across diverse device sizes.

11. Detailed Analysis of reviews.css

This section provides an in-depth examination of the reviews.css file, which governs the styling of the public-facing review section of the "Coffee's Life" web application.

11.1. General Structure and Theming

The reviews.css file **maintains a consistent brand identity** with admin.css, utilizing similar color palettes and design elements. However, it is specifically tailored for a public-facing page, focusing on clear presentation of review data and an intuitive review submission process.

11.2. Key Component Analysis

A breakdown of the styling applied to key components within reviews.css is as follows:

- **Container & Header:**
 - The .reviews-container shares a max-width of 1200px and centering with the admin panel, establishing visual consistency across the platform. It uses a cream background (var(--cream-bg, #f8f5f1)).
 - The .reviews-header employs text-align: center, features a brown h1 with a gold star icon (#ffd700), and includes a supporting paragraph, stylistically similar to the admin header but with a softer background.
- **Stats Section:**
 - The .stats-section is a clean white card designed to display overall ratings.
 - The .rating-number is presented as a large, bold, brown number (3rem), ensuring high visibility of the average rating.
 - The .star-rating-display visually represents the average rating using gold-filled (.fas) and gray-empty (.far) stars.
- **Submit Review Section (Form):**
 - The .submit-review-section is a distinct white card, clearly demarcating the review submission area.
 - The h2 heading includes a darker brown icon.
 - Form elements (input, textarea) exhibit styling consistent with admin.css, including padding, border-radius, and border-color transitions on focus, thereby maintaining a unified user experience.

- The .star-rating input provides an interactive method for users to select a star rating, with visual feedback (color change and scale transform) on hover and active states.
- A .character-count element provides useful feedback regarding review length.
- **Buttons:**
 - The .btn-primary is visually consistent with the admin panel's primary button, utilizing a **brown linear-gradient**, white text, and prominent **hover effects**.
- **Alert Messages:**
 - .alert-success and .alert-error provide clear, distinct visual feedback for form submissions, mirroring the message styles of the admin panel.
 - The .error class provides specific styling for inline validation messages.
- **Reviews List:**
 - The .reviews-section organizes individual reviews.
 - The .reviews-list uses flex-direction: column and gap for clear vertical spacing.
 - Each .review-item is a distinct card with a light border, border-radius, and a subtle transform and box-shadow on hover, making them interactive and visually appealing.
 - The .review-header employs display: flex for effective alignment of reviewer information, rating, and date.
 - Reviewer names (h4) are brown, ratings are displayed with stars (.review-rating), and dates are a softer gray.
 - review-content p ensures the readability of the review text.
- **No Reviews State:**
 - The .no-reviews class provides a friendly, centered message with a large, subtle gray icon for instances when no reviews are available.

11.3. Color Palette and Typography

The color palette and typography choices for reviews.css are as follows:

- **Primary Colors:** var(--primary-brown, #6f4e37) and var(--secondary-brown, #8b4513) are consistently used, leveraging CSS **variables** for enhanced maintainability.
- **Accent Color: #ffd700 (Gold)** is applied to stars, consistent with the admin.css file.
- **Backgrounds:** var(--cream-bg, #f8f5f1) and white are used for light, inviting sections. #fafafa is utilized for individual review items.
- **Text Colors:** var(--text-primary, #333) and var(--text-secondary, #666) ensure consistent text readability.
- **Typography:** font-family: inherit guarantees consistency with the main body font, which is 'Inter' as established in admin.css.
-

11.4. Responsive Design Approach

reviews.css also utilizes **media queries** for max-width: 768px and max-width: 480px to ensure adaptability. Key responsive adjustments include:

- General container padding adjustments.
- Scaling down of header and rating number font sizes.
- Reduction of padding across major sections (.stats-section, .submit-review-section, .reviews-section).
- The .review-header transforming from a row to a column layout on smaller screens for improved mobile display of reviewer information and date.
- Adjustment of star rating sizes for better touch targets and visual consistency on smaller screens.

12. Summary of Design Principles

Both `admin.css` and `reviews.css` adhere to a consistent and cohesive design philosophy, characterized by the following principles:

- **Branding Consistency:** A unified color palette (browns, creams, gold) and font choices ('Inter') are maintained across both administrative and public interfaces, effectively reinforcing the "Coffee's Life" brand identity.
- **Modern Aesthetic:** The application of soft gradients, rounded corners, and subtle box-shadows contributes to a contemporary and inviting visual appeal.
- **User Experience (UX) Focus:**
 - **Clarity:** Emphasis is placed on clear hierarchy in headings, well-defined sections (cards), and legible text.
 - **Interactivity:** Consistent and noticeable **hover effects** on navigation, buttons, and review items provide clear visual feedback. Interactive form elements, such as star ratings, are well-implemented.
 - **Feedback:** Standardized alert messages and status badges offer clear system feedback to the user.
 - **Readability:** Good contrast ratios and appropriate line heights (e.g., line-height: 1.6 for review content) significantly enhance overall readability.
 - **Responsiveness:** Both CSS files actively employ **media queries** to ensure optimal viewing and usability across diverse screen sizes, ranging from mobile devices to desktops. This is a critical aspect for modern web application design.
 - **Maintainability (CSS Variables):** The `reviews.css` file strategically introduces **CSS variables** (e.g., `var(--primary-brown)`) for colors. This is a best practice that facilitates easier future style modifications and promotes consistency across the codebase.

13. Architectural Observations & Best Practices (Combined)

The "Coffee's Life" web application demonstrates a strong commitment to robust development practices across both its backend and frontend:

- **Modularity & Reusability:**
 - **PHP:** Extensive use of `functions.php` and `reviews_functions.php` centralizes common logic, promoting code reuse and simplifying maintenance. Pages include these utilities as needed.
 - **CSS:** Styles are logically separated into `global.css` (for universal variables and base styles) and page-specific stylesheets. This modularity makes the CSS codebase organized, readable, and easier to manage.
- **Security:**
 - **PHP:** Critical security measures are in place:
 - **Prepared Statements:** Employed consistently for database interactions (both PDO and MySQLi) to prevent SQL injection attacks.
 - **Password Hashing:** `password_hash()` and `password_verify()` are used for secure storage and verification of user passwords, protecting against common credential-based attacks.
 - **Server-Side Validation:** Input validation is performed on the server (e.g., in `register.php`, `rating_comments.php`, `guest_checkout.php`) to ensure data integrity and security, complementing client-side checks.
- **Session Management:**
 - **PHP:** PHP sessions (`$_SESSION`) are effectively utilized to maintain user login state (`user_id`, `user_logged_in`, `is_guest`), manage shopping cart contents, and pass temporary messages (success/error alerts) across requests, providing a seamless user experience.
- **Responsive Design:**
 - **CSS:** A comprehensive and thorough approach to responsive design is evident across almost all CSS files through the extensive use of `@media` queries. Layouts, typography, and component arrangements gracefully adapt to various screen sizes, ensuring optimal viewing and usability on desktops, tablets, and mobile devices.
- **Consistent Theming & Aesthetics:**

- **CSS:** The consistent application of a "coffee" theme (specific color palettes using CSS variables, subtle background textures, coffee-related icons, and custom list styles) creates a cohesive and engaging brand identity throughout the website.
- **Animations & Transitions:** Liberal use of CSS transition properties and @keyframes animations adds a dynamic and polished feel to the user interface, enhancing interactivity and visual appeal (e.g., button hovers, element fade-ins, icon bounces).
- **User Experience (UX) Enhancements:**
 - **PHP & JS:** Client-side JavaScript (e.g., in menu.php for modal price updates, rating_comments.php for interactive stars, contact_us.php for real-time validation) combined with PHP's session-based feedback mechanism provides immediate and clear communication to the user.
 - **CSS:** Clear typography, distinct interactive element styling, and logical layouts contribute to an intuitive and pleasant browsing experience.

14. Conclusion

The "Coffee's Life" web application is a robust and thoughtfully constructed e-commerce platform. Its PHP backend effectively handles complex business logic, secure data interactions, and user state management. Complementing this, the CSS frontend provides a highly aesthetic, consistent, and fully responsive user interface. The combined implementation demonstrates a clear understanding of modern web development principles, delivering a secure, maintainable, and engaging online experience for coffee enthusiasts. The modularity and adherence to best practices in both PHP and CSS lay a solid foundation for future enhancements and scalability.

APPENDIX

Meeting No.	1
Project Title	BeanMarket Website
Date	1 June 2025
Time	12.00 p.m. – 5.40p.m.
Venue	McDonald Prima, Cyberjaya

ATTENDANCE

Present	Absent
1. Aziel Tan Zheng Chuan 2. See Wing Kit 3. Vincent Lock Chun Kit 4. Daniel Go Zi Yang	NULL

NO	ITEM	INFO/ACTION
1	1. Discuss and Decide kind of Topic and Name of Website (BeanMarket). 2. Start Preparing Report 3. Design of UI FrontEnd and BackEnd	All

The meeting adjourned at 5:45 p.m.

Prepared by:

Aziel Tan Zheng Chuan

Meeting No.	2
Project Title	Coffee's Life Website
Date	10 June 2025
Time	1.00 p.m. – 5.30p.m.
Venue	McDonald Prima, Cyberjaya

ATTENDANCE

Present	Absent
1. Aziel Tan Zheng Chuan 2. See Wing Kit 3. Vincent Lock Chun Kit	1. Daniel Go Zi Yang

NO	ITEM	INFO/ACTION
1	1. Changed the name of website from 'BeanMarket' to 'Coffee's Life'. 2. Design Admin FrontEnd and BackEnd 3. Finding and adding interesting feature in our Website	All

The meeting adjourned at 5:30 p.m.

Prepared by:

Aziel Tan Zheng Chuan

Meeting No.	3
Project Title	Coffee's Life Website
Date	21 June 2025
Time	11.00 p.m. – 5.30p.m.
Venue	Chagee Meranti Puchong

ATTENDANCE

Present	Absent
1. Aziel Tan Zheng Chuan 2. See Wing Kit 3. Vincent Lock Chun Kit 4. Daniel Go Zi Yang	NULL

NO	ITEM	INFO/ACTION
1	1. Testing website 2. Fix bugs 3. Add Detail and effect into the website 4. Finishing the report	All

The meeting adjourned at 5:30 p.m.

Prepared by:

Aziel Tan Zheng Chuan

Meeting No.	4
Project Title	Coffee's Life Website
Date	26 June 2025
Time	10.00 p.m. – 4.30p.m.
Venue	McDonald Prima, Cyberjaya

ATTENDANCE

Present	Absent
1. Aziel Tan Zheng Chuan 2. See Wing Kit 3. Vincent Lock Chun Kit 4. Daniel Go Zi Yang	NULL

NO	ITEM	INFO/ACTION
1	1. Finished report 2. Final Check of the overall Report and Code	All

The meeting adjourned at 4:30 p.m.

Prepared by:

Aziel Tan Zheng Chuan