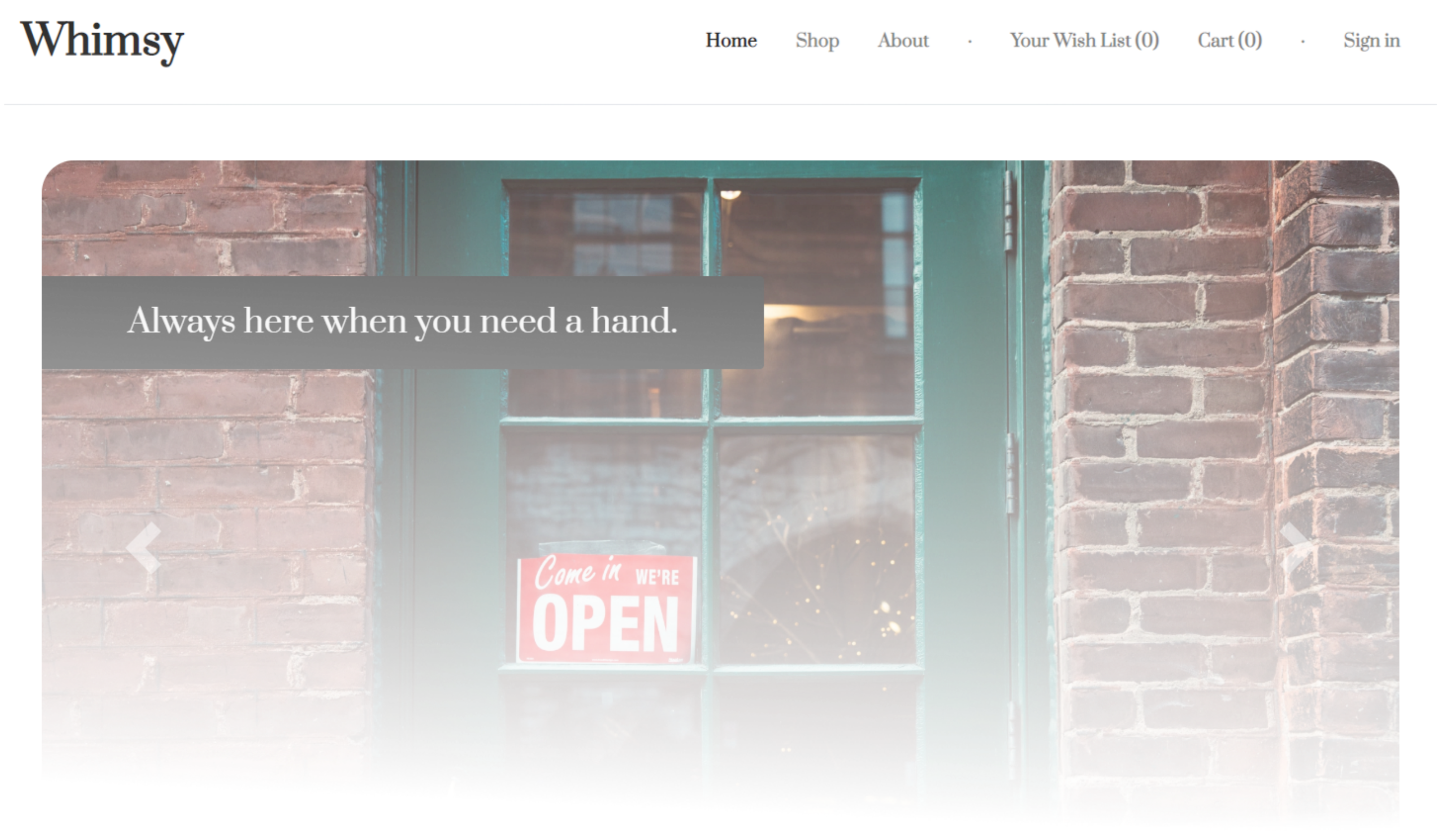
Whimsy: An Online Store

By: Abel Worku



Project Whimsy: Documentation

By: Abel Worku

Table of Contents:

Introduction

This document will serve as the future home for all related project documentation. It is intended to organize all documentation related to the project, including elements such as possible improvements for the project, ongoing features being built, and explanations of all components contributing to the functionality of the project.

Functionalities Explained

The functionalities implemented for the project include those that perform certain background tasks for the user when they are on the website, such as: handling the sign-in, sign-up, and sign-out process, handling asynchronous AJAX requests, modifying session state variables related to user activity, and querying the database to retrieve any required information. Below are the associated php pages which perform these tasks as well as others visible to the user which contribute to the functionality of the website:

Front-End Components

Home.php

- The website's main landing page for the user, and introduces the user to the website as an online store by detailing its mission and intentions.
    - A slideshow introduces mission-like statements for the user and guides them to the main shop page
- Includes links to a "Shop by Category" row, where the user can choose from all categories of products and be sent to the main shop page (Shop.php) with a pre-customized search for their chosen category
- Includes links to separate pages providing informational content to the user, such as an about page (About.php), a team page (Team.php) a frequently asked questions page (FAQ.php), and a contact us page (ContactUs.php)
- The homepage's footer provides the user with the website's core mission statement, and basic contact information

Shop.php

- The website's main browsing page and shop catalogue page for searching through all the products. Search functionality is implemented in the form of three terms - category, price, and rating (where each product contains this information in the database by linking between tables and using the database's (shopDB.sql) table relationships) - where they can be combined to form a more specific search or used separately for a broader search as needed to reflect user preferences when browsing.

- Displays products as a set of cards numbering three per row (one card for each product), and can be resized depending on the browser and display size to suit a more responsive design (and mobile-friendly interface)
    - On a medium-sized display, the shop page's front-end (with the Bootstrap CSS framework) will display a set of cards numbering 2 per row, and on a smaller-sized display, the shop page's front-end will display a set of cards numbering 1 per row
- In one card representing a product, the user may:
    - Go to the product's own page (ProductPage.php) with a highlightable "Shop" button (or with the product's image and title)
    - Add the product to the user-related session's wish list with a highlightable "Add to Wish List" button
        - This is done by sending an asynchronous GET request using jQuery and AJAX to the back-end page addItemToWishList.php, containing the product's main identifier in the database, the productID (a field in the Products table). The productID is used because it can provide an SQL query a main starting point for querying related information about the product (such as its name, its description, its shipping speed, and its return policy), which will be useful in this regard for the page WishList.php when it needs to query more information about a product
            - If the asynchronous GET request was successful, a popover will appear above the button notifying the user that their item was successfully added to the wish list
        - Thereafter, addItemToWishList.php will receive the productID and add it to the wish list's session state variable $_SESSION['wishList'], stored in PHP's superglobal associative array $_SESSION for maintaining session state in a website
            - $_SESSION['wishList'] stores items as a key-value associative pairing between the key "productID" and the value productID received by a GET request
        - In addition, the top navigation bar provided by header.php will dynamically update upon receipt of new information about the size of the wish list (by counting the number of items in $_SESSION['wishList']), after a refresh of the page or by visiting a new page
            - Ex: Displays the current number of items in the wish list as "Your Wish List (1)"
    - Add the product to the user-related session's cart with a highlightable "Add to Cart" button
        - This is done by sending an asynchronous GET request using jQuery and AJAX to the back-end page addItemToCart.php, containing the

product's main identifier in the database, the productID (a field in the Products table). The productID is used because it can provide an SQL query a main starting point for querying related information about the product (such as its name, its description, its shipping speed, and its return policy), which will be useful in this regard for the page Cart.php when it needs to query more information about a product
        - If the asynchronous GET request was successful, a popover will appear above the button notifying the user that their item was successfully added to the wish list
    - Thereafter, addItemToCart.php will receive the productID and add it to the cart's session state variable $_SESSION['cart'], stored in PHP's superglobal associative array $_SESSION for maintaining session state in a website
        - $_SESSION['cart'] stores items as a key-value associative pairing between the key "productID" and the value productID received by a GET request
    - In addition, the top navigation bar provided by header.php will dynamically update upon receipt of new information about the size of the cart (by counting the number of items in $_SESSION['cart']), after a refresh of the page or by visiting a new page
        - Ex: Displays the current number of items in the cart as "Cart (1)"

About.php

- Informs the user of the main intentions of the website, its objectives, its mission, as well as its theme to allow the user to better understand the store

WishList.php

- Provides a place for the user to save items they find to a dedicated space of the website, as a possible junction to initiating the checkout process by adding the items to the cart
- Displays all user-added wish list items and a count of how many items are currently in the wish list (i.e. how many key-value pairs exist in the session state variable for the wish list $_SESSION['wishList'])
- Options are provided to the user to remove a specified item from the wish list (as a button named "Remove" in the product's card), and to clear the wish list of all items (as a button named "Clear All" at the top of the wish list)
- Products are displayed similarly to cards from the website's shop page Shop.php, with relatively less information in order to convey the most important elements about the product to the user - the product's image, the product's title, and its price, category, and tag (the user can also navigate back to the product's page if desired with the product's image and title)

- If the wish list is empty (i.e. no items added to the session state variable for the wish list $_SESSION[‘wishList’]), a special message will be displaying informing the user with a guide back to the website’s shop page (Shop.php) to browse for items
- Interacts with removeItemFromWishList.php

Cart.php

- Provides a place for the user to save items they intend to purchase to a dedicated space of the website
- Displays all user-added cart items, a count of how many items are currently in the cart (i.e. how many key-value pairs exist in the session state variable for the cart $_SESSION[‘cart’]), and a total cost of all the items by calculating the sum of the prices of all the products (before the user purchases them in a hypothetical extended checkout process)
- Options are provided to the user to remove a specified item from the cart (as a button named “Remove” in the product’s card), and to clear the cart of all items (as a button named “Clear All” at the top of the wish list)
- Products are displayed similarly to cards from the website’s shop page Shop.php, with relatively less information in order to convey the most important elements about the product to the user - the product’s image, the product’s title, and its price, category, and tag (the user can also navigate back to the product’s page if desired with the product’s image and title)
- If the cart is empty (i.e. no items added to the session state variable for the wish list $_SESSION[‘cart’]), a special message will be displaying informing the user with a guide back to the website’s shop page (Shop.php) to browse for items
- Interacts with removeItemFromCart.php

SignIn.php

- Displays two forms for the user, with the leftward form for signing in with an existing account and the rightward form for signing up with a new account
- Allows the user to sign up with a new account, or sign in with an existing account.
    - To sign up with a new account, the user provides a new username and password, and then goes to the “Sign up” button below. Thereafter, an asynchronous POST request is sent to the page SignUpService.php containing the username and password provided from the input form, to perform the sign up process
    - To sign in with an existing account, the user provides a username and password which must already exist in the database, and then goes to the “Sign in” button below. Thereafter, an asynchronous POST request is sent to the page SignInService.php containing the username and password provided from the input form, to perform the sign in process

ProductPage.php

- Displays a large image of the product and detailed information about the product, including average rating, shipping speed, return policy, current inventory, and more.
- Above the product is a large title with the product's category for user reference (and is also listed below in the detailed information section for the product)
- Below the product's extended information are two rows of product listings recommending the user more products based on a shared category or shared tag (an additional descriptor of a product, stored in the database for each product), and a third row displaying a list of recently viewed items (by accessing the session state variable array for recently viewed items, $_SESSION['recentlyViewed'])
  - The third row displaying a list of recently viewed items maintains the session state variable array $_SESSION['recentlyViewed'] to a maximum of 4 items, in order to display a row of 4 items and not a row of items extended to the size of the session state variable for the list of recently viewed items.
- Interacts with addItemToWishList.php and addItemToCart.php to add items to the wish list and cart, with the same process as described for Shop.php

header.php

- A separate navigation bar that each front-end page for the website links to, to display up-to-date information related to the user session.
  - This includes the number of items currently in the wish list and cart (displayed as "Your Wish List (1)" and "Cart (1)" respectively), and if the user is signed in, a greeting to the user with their username and an option to sign out (if the user is not signed in, there will instead be a sign in link named "Sign In")

Back-End Components

SignUpService.php

- Responds to a POST request from form input received from the website's sign in page SignIn.php, after the user completes the username and password field to sign up (with a new account). SignUpService.php will take the retrieved username and password, and will hash the password to prevent it from being plainly exposed (using PHP's built-in function "password_hash()") allowing for safe and protected storage in the database. After hashing the password, both the username and hashed version of the received password will be added as a record in the "Users" table of the database (with two fields "Username" and "Password")
- SignUpService.php will then initialize a session state variable $_SESSION['userName'] to store the username (indicating a newly created user session and that the user has signed in, to identify the user for their current session,

as well as to personalize the user experience when browsing through the website), and redirect to the home page Home.php for introductory page navigation

SignInService.php

- Responds to a POST request from form input received from SignIn.php, after the user completes the username and password field to sign in (with an existing account in the database shopDB). If the user enters a username which already exists in the database, and their password is matched successfully to the corresponding entry in the database (using the PHP built-in function “password_verify” to compare the user-provided password with the protected hashed password), SignInService.php will then initialize a session state variable $_SESSION[‘userName’] to store the username (indicating a newly created user session and that the user has signed in, to identify the user for their current session, as well as to personalize the user experience when browsing through the website), and redirect to the home page Home.php for introductory page navigation

SignOutService.php

- Clears all session-related information identifying the user, including the session state variables for the wish list, the cart, the user’s recently viewed items, and the user’s username ($_SESSION[‘wishList’], $_SESSION[‘cart’], $_SESSION[‘recentlyViewed’], and $_SESSION[‘userName’] respectively). Thereafter it will redirect to the SignIn.php page as a post-session landing page for the user

addItemToWishList.php

- Responds to an asynchronous GET request from a shop-related link (ex. an “Add To Wish List” button in the shop page Shop.php or the product’s page ProductPage.php), requesting to add the item to the website’s wish list (i.e. the website’s session state variable responsible for storing wish list items, $_SESSION[‘wishList’]). It receives the query string parameter “productID”, which is a product identifier uniquely identifying the product in the database (and allowing for further SQL queries when more information about the product is required), and adds the “productID” identifier as a key-value pair to the session state variable for the wish list (which is an array)

addItemToCart.php

- Equivalent in functionality to addItemToWishList.php, with the difference that it modifies the session state variable responsible for storing cart items, $_SESSION[‘cart’]

removeItemFromWishList.php

- Responds to a synchronous GET request from the wish list page WishList.php (through the query string parameter "productID" in the URL sent from WishList.php, visiting the page removeItemFromWishList) that it services, and uses the given product identifier information "productID" to remove the associated product from the wish list session state variable. If a GET request has not been found to be set, removeItemFromWishList.php will empty all key-value pairs in the session state variable for the wish list, and redirects back to the wish list page WishList.php.

removeItemFromCart.php

- Equivalent in functionality to removeItemFromWishList.php, with the difference that it modifies the session state variable responsible for storing cart items.

Possible Improvements

Possible improvements to the project for the future would include implementing components of the project that had to be removed for the sake of a lower priority, but given enough time would still enhance the project as a whole. I would wish to add more functionality to the Shop page's search mechanism, and allow the user to search more freely with a custom search of user-specified terms, rather than limiting it to the combination of the three search terms "Category", "Price", and "Rating". As well, expanding the tracking operations of the user's activity to better recommend the user products and information would be a significant asset to the project (for example, monitoring the frequency of a user's visits to a certain category or tag of products and customizing a recommended products space to reflect that aspect of the user's activity).

Ongoing Features

Web Technologies used in Development

The web technologies used in the project development are: HTML, CSS, Bootstrap, JavaScript, jQuery, AJAX, PHP, and MySQL for the database and querying the database. The LAMP stack was used to operate on a local server, using the XAMPP application.

Justification of Web Technologies

Front-End

HTML and CSS were used as the main starting points for developing the project's front-end, however during the early phases of the project it became clear that using a CSS framework would abstract away many time-consuming issues related to deploying the front-end quickly (with speed in mind to allow for time to develop the website's functionality and back-end). I was familiar with HTML and CSS by practicing with this course's assignments, but was not familiar with Bootstrap. As a result, I read through relevant documentation, learning resources, and practice material in order to gain an understanding of the framework's core ideas. With this I was able to design the front-end more efficiently and effectively, while being able to preserve and enhance the design goals I had in mind from the beginning of the project (such as ease of use and responsiveness). The decision to use the Bootstrap framework was related to a helpful suggestion from the teacher, as in the early phases of the project many pages were planned for the project prototype that could be better implemented with a framework. In addition, JavaScript, jQuery, and the AJAX web technologies were chosen to be used in large part to the familiarity that I had with them during the course, and I believed it would make the development process more efficient to rely on technologies which I had a prior understanding of.

Back-End

PHP and MySQL were also selected to be used with familiarity in mind, as I had an understanding of their capabilities and what could be accomplished with them, allowing for the project development to not be hindered by components that might not be realistically implemented to a desired standard. When planning the project's development, I had the intention for designing the website to be able to communicate with its back-end, communicate with the database, and be able to interact with particular sessions, and PHP and MySQL appeared to be the most favourable for those tasks (in conjunction with my prior familiarity with them).

User Guide

The store was designed with simplicity in mind to allow the user to navigate the website more efficiently. The user may first arrive at the home page, and is greeted with an introduction about the website. While on the homepage they may scroll down to a row of cards linking directly to the Shop page with a pre-customized search to search for a certain category of products.

When first arriving on the home page, the slideshow can be explored by clicking the left or right arrows. In the middle of the home page, there is a "Shop by Category" row where one can link directly to the Shop page with a pre-customized search for a certain category of products, by clicking and highlighting a category card.

In the Shop page, one can search through the store's catalogue of products with a combination of terms identifying certain product information - category, price, and rating. The search can be customized with these terms, and one can click the "Search" button to perform the search - relevant results of products are displayed below. A list of products with equally sized cards are displayed, and functions are offered to add it to the wish list, or add it to the cart, or to open the product in a separate product page for more information.

The product page displays all related information for a product, and one can add the product to the wish list and the cart in this page as well. The user can also scroll down to see related products by category or tag, and any products they have viewed recently in their current session.

After items have been added to the wish list and cart, the user can view their wish list and cart by using the navigation bar at the top of any page of the website. The "Your Wish List" page displays the items in the wish list, how many items there are, and options to delete separate items or all items altogether. The "Cart" page operates similarly, and also displays a total cost of all the items as one order.

The user can sign in to the website with an existing account, or sign up with a new account by going to the "Sign in" link, where they are guided with two forms titled "Sign in" and "Sign up" respectively. After signing in or signing up successfully they are redirected back to the home page with their username displayed at the top of every page.

How to Open the Application

Locally

To run the application locally, one may use the LAMP software stack or a variation of it, such as XAMPP (XAMPP will be discussed here). The LAMP software stack consists of Linux, Apache, MySQL, and PHP which together provide a platform to support running this application on. XAMPP, a LAMP-like software stack, consists of Apache, MariaDB (similar to MySQL), PHP, and Perl. Because this application relies on PHP, and PHP is a server-side programming language, communicating with a server to process requests is necessary for its intended functionality - the XAMPP stack allows for the application to communicate with a local server (via HTTP) to perform server-side tasks (such as retrieving data from a database or adding data to a database) with tools such as PHP (Apache server), to have the capability to interact with relational databases with a relational database management system (MySQL), and to have the capability to build a dynamic website that responds to server-side processes (PHP).

Here are a set of instructions to guide you to setting up XAMPP on your local device, and then running the application locally:

- Download XAMPP (XAMPP is free to download)
- After installing it, open XAMPP. A window titled XAMPP Control Panel should appear. There are five rows at the top of the window, with the first two named "Apache" and "MySQL" - across from them are four actions named "Stop", "Admin", "Config", and "Logs". To start Apache running as a local server, and to start MySQL, go to the "Start" button in the "Actions" section of their respective rows.
- Locate XAMPP's installation destination on your local device, it should appear as the folder "xampp" in your C:/ drive. Then go to the folder "htdocs", and place a copy of the application into htdocs so that the application can be run on the local server provided by XAMPP.
- The application can now be accessed via any browser with the URL "http://localhost/CP476-Internet-Computing-Final-Project/HTML/Home.php", which takes you to the homepage of the application, Home.php. From here you may navigate as desired throughout the website, such as the Shop page to browse through items as well as creating an account and being able to sign in with the account.
    - If you would also like to see the components of the application, including its back-end pages as described in the documentation, you may open the files from the copy of the application you have in a text editor, such as Notepad++ or Brackets, or you may visit this URL "http://localhost/CP476-Internet-Computing-Final-Project/" for a main index of all the files related to the project, found in index.php

On the Web

This application is intended to be accessible on the web through a web hosting service such as 000webhost or Heroku. At the moment, it is hosted on 000webhost at this URL "https://whimsy.000webhostapp.com/HTML/Home.php", and also "https://whimsy.000webhostapp.com/" for a main index of all the files related to the project.

Contact Information