TECHNICAL DOCUMENT

DATABASE SET UP

In setting up the database for the project, phpMyAdmin was the chosen tool due to its ease of use and accessibility, which was particularly beneficial given my prior experience with it. Its compatibility with PHP made it an intuitive choice for the web application development, ensuring a smoother integration and management process. Furthermore, when compared to MySQL Workbench, phpMyAdmin's simpler interface was more suited to my needs, providing a less complex and more straightforward user experience. This decision significantly contributed to a more efficient and manageable database setup phase.

The database for the project is structured around a relational model, specifically designed to cater to the needs of an e-commerce platform. This structure includes interconnected tables that efficiently manage and represent various entities such as users, roles, products, and orders. This relational design is fundamental in facilitating complex queries and operations, allowing for a robust and scalable system that can handle the intricate relationships and data flows inherent in e-commerce activities.

TABLES AND THEIR ROLES:

role	Manages different user roles within the application, defining permissions and access levels. Columns include the role name, details about the role, a deleted column to indicate if role has been deleted, and a unique ID for each role.
user	Stores personal and account information for users, essential for login credentials and profile management. Includes personal information (email, name, surname), account information (when they joined, password), contact number, a foreign key to the user's role, a deleted column, and a unique user ID.

brand	Lists available brands on the platform, helping users identify products from their preferred manufacturers. Consists of the brand name, details, an image URL, a deleted column, and a unique brand ID.
category	Categorizes products, making it easier for users to browse and find items of interest. Contains a category name, details, a deleted column, and a unique category ID.
product	Contains detailed information about products for sale, including descriptions, pricing, and stock levels. Includes product name, description, foreign keys to its category and brand, price, stock levels, an image URL, a deleted column, date added, and a unique product ID.
wishlist	Tracks user wishlists, allowing users to save and revisit desired products. Has foreign keys to the user and product tables, and a unique wishlist ID.
order_product	Acts as a junction between products and orders, detailing which products are included in each order. Contains foreign keys to both the order and product tables.
orders	Records and details customer orders, tracking everything from creation to delivery status.

	Includes when the order was created, updated, foreign keys to the order status and user's address, a deleted column, and a unique order ID.
orderstatus	Keeps track of various order statuses, providing updates on order progress from placement to delivery. Contains the status description, a deleted column, and a unique status ID.
address	Manages user address information, crucial for shipping and billing processes. Includes address details (street, city, ZIP code, region), a foreign key to the associated user, an indicator if it's the default address, a deleted column, the name and surname associated with the address, a mobile number, and a unique address ID.

DATA TYPES:

varchar	Accommodates text data, such as names, email addresses, and product descriptions, with a specified maximum length.
datetime	Records dates and times, essential for tracking events like order placements and user registrations.
tinyint	Often used for small numerical data and status codes.
int	Stores larger numerical values, commonly used for identifiers like user IDs and order numbers.

longtext	Holds extensive text data, suitable for lengthy descriptions that exceed the typical length limitations of standard text fields.
float	Accommodates numbers with decimals, used for data like prices and quantities where precision is important.

RELATIONSHIPS BETWEEN TABLES:

The database incorporates both one-to-many and many-to-many relationships to capture the interactions and connections among various entities. These relationships are vital to ensure data integrity across the application.

Relationships between tables are established using foreign keys. A foreign key is a field (or collection of fields) in one table that uniquely identifies a row of another table or the same table. In my database, the foreign keys create a link between related data across different tables, allowing for relational database operations such as joins. Here's how the relationships are established:

User and Role	The 'Role' field in the 'user' table is a foreign key that references the 'ID' field in the 'role' table. Many-to-one relationship between users and roles.
Product, Brand, and Category	The 'Brand' field in the 'product' table is a foreign key that references the 'ID' field in the 'brand' table, linking each product to a specific brand. The 'Category' field in the 'product' table is a foreign key that references the 'ID' field in the 'category' table, indicating which category a product belongs to.

	One-to-many relationship with both categories and brands.
Orders, User, Address, and Order Status	The ‘user’ field in the ‘orders’ table is a foreign key that references the ‘ID’ field in the ‘user’ table, establishing which user placed the order. The ‘address’ field in the ‘orders’ table is a foreign key that references the ‘ID’ field in the ‘address’ table, indicating the shipping address for the order. The ‘status’ field in the ‘orders’ table is a foreign key that references the ‘ID’ field in the ‘orderstatus’ table, signifying the current status of the order.
Wishlist and User-Product	The ‘user’ field in the ‘wishlist’ table is a foreign key that references the ‘ID’ field in the ‘user’ table, identifying the user who has a given wishlist. The ‘product’ field in the ‘wishlist’ table is a foreign key that references the ‘ID’ field in the ‘product’ table, denoting the products that are in a user's wishlist. many-to-many relationship between users and products through the wishlist.
Order Product, Orders, and Product	The ‘orderid’ field in the ‘order_product’ table is a foreign key that references the ‘ID’ field in the ‘orders’ table, linking the ordered products to a specific order. The ‘productid’ field in the ‘order_product’ table is a foreign key that references the ‘ID’ field in the ‘product’ table, connecting the order with the products that have been ordered. many-to-many relationship between orders and products through order_product.

Address and User	The ‘User’ field in the ‘address’ table is a foreign key that references the ‘ID’ field in the ‘user’ table. This sets up a relationship where multiple addresses can be associated with a single user (for instance, billing and shipping addresses). one-to-many indicating that the user can have multiple addresses.

DATABASE INSTALLATION AND CONFIGURATION PROCEDURE

Step 1:	Downloading and Installing XAMPP from the official Apache Friends website.
Step 2:	Launching XAMPP on Windows, Accessed the XAMPP Control Panel through the Start Menu.
Step 3:	Starting Modules Apache and MySQL from the XAMPP Control Panel.
Step 4:	Opened Web browser and visited 'http://localhost/' to confirm that Apache was running. Accessed 'http://localhost/phpmyadmin/' to manage MySQL databases using phpMyAdmin. Created a PHP file in the 'htdocs' directory and it was accessed via 'http://localhost/test.php' to verify that PHP was functioning correctly.
Step 5:	Database Setup

Step 5.1:	I accessed phpMyAdmin and located the menu on the left side. I then clicked on 'New' to start creating a new database. I named the database and clicked 'Create.'
Step 5.2:	I selected the newly created database from the side menu, which loaded its structure page for further configuration.
Step 5.3:	I began creating the first table within the database. I gave the table a name and specified the required number of columns, then clicked "Create" to establish it.
Step 5.4:	After the table was created, a form appeared with various inputs. For each column, I entered a name, chose the data type, defined the length/values. Then I determined whether the column could be NULL, Auto Increment (A.I), etc. If I needed more columns, I used the option near the table name to add them. After configuring all columns, I saved the table.
Step 5.5:	Following the same procedure as in Step 5.3, I created any additional tables required for the website, repeating the process for each table.
Step 5.6:	To define relationships between tables, I navigated to the 'Relation view' in phpMyAdmin and used this feature to create foreign keys and establish the necessary table relationships.

In terms of security and user roles, I defined three distinct user roles: Administrator, Employee, and Customer. The Administrator role granted full access to the entire system, ensuring administrative oversight and management capabilities. On the other hand, the Employee role had limited access to the database, allowing employees to interact with specific data without extensive administrative rights. Lastly, the Customer role was designed without direct database access, aligning with the common user experience where direct database interaction is typically unnecessary and carries potential security risks.

DATA MANIPULATION TECHNIQUES

CRUD OPERATIONS

This section concisely outlines the Create, Read, Update, and Delete (CRUD) operations in a PHP-based web application. It's essential for managing data across various entities such as users, roles, products, brands, categories, and orders.

CREATE

createUser

createRole

createBrand

createCategory

createProduct

1. Inputs are collected and sanitized to prevent SQL injection, ensuring data security and integrity.
2. Constructs the query: INSERT INTO table_name (column1, column2, ...) statement to define the data insertion process.
3. Executes the INSERT INTO SQL statement using prepared statements with the sanitized data to insert data into the corresponding tables.
4. Provides feedback indicating the success or failure of the insertion to inform the application and its users of the operation's outcome.

READ

GetUsers

GetRoles

GetBrands

GetCategories

GetProducts

1. A SELECT query is formulated based on the need, such as SELECT * FROM table or SELECT * FROM table WHERE condition, tailoring it with specific conditions to retrieve the desired records.
2. Executes the SELECT query, often using prepared statements if variables are involved, to securely fetch the data.
3. The data is returned as an array of results or a single result set, facilitating further manipulation or display.

UPDATE

updateUser

updateRole

updateBrand

updateCategory

updateProduct

1. The record's existence is verified to ensure that the update is applicable.
2. Inputs are sanitized to maintain data integrity and prevent security issues.
3. Constructs the query: UPDATE table_name SET column1=?, column2=? WHERE condition statement to define how the records will be modified.
4. Executes the prepared UPDATE statement with the new, sanitized data to modify the records.
5. Provides feedback to indicate the success or failure of the update operation.

DELETE

‘deleteUser’

‘deleteRole’

‘deleteBrand’

‘deleteCategory’

‘deleteProduct’

1. Confirms that a soft delete is the appropriate operation to maintain data integrity and allow for potential data recovery.
2. Constructs the query: ‘UPDATE’ table_name SET deleted = 1 WHERE condition’ statement to specify the soft delete process, where ‘deleted’ is a column indicating the active/inactive status of the record.
3. Executes the prepared ‘UPDATE’ statement with the necessary conditions to accurately target and mark the intended record(s) as inactive.
4. Provides feedback confirming the success or failure of the soft delete operation to inform the application and its users about the outcome.

ORDERS MANAGEMENT

CREATE	CreateOrder() Handles additional complexities like user details and inventory adjustments. Includes steps to handle order specifics like totals, user details, and potentially inventory adjustments.
READ	GetOrders() Retrieves comprehensive and order details, including data from joined tables such as users, products and addresses.
UPDATE	UpdateOrder() Updates and Adjusts order details, status, or other related information based on user or administrator's actions.
DELETE	DeleteOrder() Involves setting the order as cancelled rather than fully deleting it.

ADDITIONAL FUNCTIONS

getGUID	Generates a unique identifier for certain operations.

adminLogin()
 	
userLogin()
	Authenticate users, these functions are used to check provided credentials against the database and manage session creation.

DATA VALIDATION AND OPTIMIZATION

User Login	Upon receiving a POST request for user login, the script validates the input by sanitizing the email and password using htmlspecialchars and addslashes. This process helps mitigate potential XSS and SQL injection attacks. The sanitized credentials are then passed to the userLogin function, which verifies the user's identity against the database. Upon successful authentication, the user is securely redirected to their account page, ensuring a seamless and secure login process, else if the user is not found in the database or the information inputted is incorrect, the user is shown an error message.
User Registration	For new account registrations, the script first checks the existence of the email inputted by _POST. It then proceeds to sanitize the input to prevent malicious data entry. A check is performed to ensure the user doesn't already exist in the database to avoid duplicate entries. New users are created using the createUser() function, with passwords being securely hashed using SHA-1.
Session Management	The session management script is invoked during user logout. It resets and clears all session data, ensuring no sensitive information is left behind. The session is then destroyed, and a new session ID is generated for subsequent interactions. Users are then redirected to the homepage, effectively logging them out and safeguarding their session data.
Form Submission	The application handles various types of form submissions with dedicated logic to validate and process the data. This includes updating user details, changing settings, or accessing restricted pages. Each form submission undergoes thorough sanitization and validation checks to ensure the data is accurate and secure before any processing or database interaction occurs.

Data Sanitization	htmlspecialchars and addslashes are used all through to prevent XSS and mitigate certain forms of SQL injection to ensure that data displayed back to the user or sent to the database is free from malicious scripts or characters.

VIRTUAL SERVER CONFIGURATION

VIRTUALIZATION SOFTWARE

XAMPP has been chosen as the preferred virtualization software for creating a local server environment, primarily due to its integrated package of Apache and MySQL. This suite, which includes access to PHP and phpMyAdmin, is specially designed for the development of PHP-based web applications and efficient database management. The all-in-one nature of XAMPP significantly reduces setup complexity, allowing developers to concentrate more on coding and less on configuration.

Its compatibility across Windows, Linux, and macOS ensures a flexible setup suitable for various operating systems, making it a versatile choice for different development scenarios. The substantial user base of XAMPP offers extensive community knowledge and support, facilitating troubleshooting and learning through readily available shared experiences and solutions.

As a free and open-source solution, XAMPP is also cost-effective, appealing to individuals or teams operating within tight budgets. The installation, setup, and configuration details of XAMPP have been previously covered in the database installation and configuration section, highlighting its integral role and the necessity of both components for functional operation.

DYNAMIC WEB APPLICATION DEVELOPMENT

FRONTEND TECHNOLOGIES:

HTML (HyperText Markup Language) is used as the backbone of the web application, defining the structure of web pages with elements like headers, paragraphs, links, and forms.

CSS (Cascading Style Sheets) is used to style of the website which can be changed without altering the HTML, enhancing the visual appeal of the application, making it more engaging and accessible to users. It also improves the navigation and readability, contributing to a positive user experience.

Bootstrap is a set of pre-designed components and a grid system that speeds up the development process by providing a consistent framework for the layout which help ensure the application is responsive on a variety of devices and screen sizes, from mobile phones to large desktop monitors. This responsiveness is key to user satisfaction in our multi-device world.

JavaScript enables interactive features on web pages, such as form validation, real-time content updates, and interactive maps, enabling a dynamic and responsive user interface that reacts to user inputs without requiring page reloads, which can greatly enhance the perceived speed and responsiveness of the application. jQuery is a JavaScript library making it quicker and easier to develop JavaScript functionality.

Tab Switching in the UI	‘buttonSwitch(btnClicked)’ Handles the switching between login and registration buttons. It toggles the display and active status of buttons based on the user's selection. Used in the createAccount page where a dual-form interface is implemented which enables users to choose whether to log in or register.
Quantity Selector for Cart Items	Increment and Decrement Buttons Allow users to increase or decrease the quantity of an item in their shopping cart. It dynamically updates the total price based on the new quantity. Clicking the increment/decrement button triggers a change in the quantity field, updates the session's cart quantity through a POST request to 'updatecartquantity.php', and recalculates the subtotal and total prices displayed on the page.

Account Navigation Handling	‘$('#accountOptions').on('click', ...)’ Manages the active state of account navigation options. It ensures that only one option is active at a time and updates the display accordingly. Used in the Account Page, where the user is able to navigate through different sections of the page: addresses, details, orders, overview and wishlist.
Navigation to Account Sections	‘navigateToWishlist()’ and ‘navigateToOrders()’ These functions redirect the user to specific parts of the account management page (wishlist or orders section) and ensure the page scrolls to the appropriate section upon loading. Used in the Account Page, as it does not consist of different pages, it is required to use this method to open different sections.
Address Modal	‘setModalAddressFields()’ and ‘clearModalAddressFields()’ Fills in the address details in a modal when editing an existing address or clears the fields when adding a new address. It ensures that the modal reflects the correct information for the user's actions. Used in the Account Addresses section.
Add to Cart Functionality	‘setValToAddToCart(guid)’ Sets a value indicating a product should be added to the cart and then submits the form. This is part of the process of adding items to the user's shopping cart.

	Used wherever the 'Add to Cart' button is found.
Modal Fill Field	• ‘setModalRoleFields(id, name, details)’ • ‘setModalBrandFields(id, name, details)’ • ‘setModalCategoryFields(id, name, details)’ • ‘setModalOrderStatusFields(id, name)’ • ‘setModalProductFields(id, name, description, shortDescription, category, price, stock, brand)’ • ‘setModalUserFields(id, name, surname, email, number, role, password)’ Fills the modal's fields with existing data for a specific role.
Modal Field Clearing	• ‘clearModalRoleFields()’ • ‘clearModalBrandFields()’ • ‘clearModalCategoryFields()’ • ‘clearModalOrderStatusFields()’ • ‘clearModalProductFields()’ Clears all input fields in the respective modal, ensuring a clean slate for new entries.

BACKEND TECHNOLOGIES:

PHP (Hypertext Preprocessor) is a server-side scripting language tailored for web development, playing a pivotal role in handling server-side logic and database interactions essential for dynamic web applications. Embedded within HTML, it manages dynamic content, databases, and session tracking, and can construct extensive e-commerce platforms. PHP scripts, running on the server, generate HTML content dynamically in response to user actions and database queries, processing input from HTML forms and interacting with databases to reflect real-time changes. It efficiently performs CRUD operations, manages user sessions, and handles authentication, ensuring a seamless and personalized user experience by directing users to appropriate pages post-interaction.

PHPMyAdmin, a PHP-based tool, is instrumental in administering MySQL databases, providing a user-friendly interface for intricate database management tasks. Although it doesn't have a direct impact on the frontend, it's essential for developers, allowing the execution of SQL queries through PHP, which in turn influences the HTML content presented to users. This functionality, combined with PHP's server-side capabilities, ensures data is valid, sanitized, and user interactions are both dynamic and secure, significantly enhancing the overall application's performance and reliability.

USER INTERACTION

Navigation	Users interact with the application through menu items, buttons, and links created using HTML and styled with CSS and Bootstrap. These navigational elements lead users to different pages or sections, facilitating easy movement throughout the site.
Forms	Forms enable users to submit data, including login credentials, search queries, or new content. JavaScript and jQuery are utilized for client-side form validation, ensuring data is correct and complete before submission.
Interactive Components	Bootstrap's interactive components, such as modals, dropdowns, carousels, and tabs, enhance user engagement. JavaScript and jQuery activate and manage these elements, providing a dynamic and interactive user experience.

DYNAMIC RESPONSES AND RENDERING

Server-Side Rendering	When a user submits a form or interacts with the application, the request is sent to the server where PHP processes it. Depending on

	the interaction, PHP might render a new page displaying a confirmation, error message, or a personalized dashboard.
Data Display and Updates	After interactions requiring data retrieval, such as searching for products or viewing profiles, PHP queries the database and renders a page with the requested information. If the user edits their information, PHP updates the database and renders a page reflecting the changes.
Client-Side DOM Manipulation	JavaScript, with the help of jQuery, can dynamically change the content displayed on the page after it has loaded. For example, it can show or hide elements, display validation errors, or update styling in response to user actions.
Feedback and Alerts	Bootstrap's alert system, controlled via JavaScript, provides immediate feedback to users. If a user attempts to submit a form with incomplete information, an alert might inform them of the necessary corrections.
Content Organization	Bootstrap's grid system and components like pagination, tabs, and accordions allow users to interact with and manage large sets of data or content efficiently, maintaining a clean and organized user interface.
Session-Based Interactions	PHP manages sessions to track user activity across page loads, essential for user authentication and maintaining a continuous experience. It can personalize the interface by displaying user-specific information and preferences.
Full Page Reloads for Updates	To reflect changes following user interactions, the application uses full page reloads. For instance, after submitting a form or updating information, the page reloads to show the latest content or feedback.

	This ensures users always have the most current information and a clear understanding of the result of their actions.

TEST CASES

FUNZIES – TEST CASES

ID	Test Case	Steps	Expected Result	Actual Result	Status
TC01	Add to Cart Functionality	1. Ensure the product has a unique ID visible to the system. 2. User clicks on the "Add to Cart" button or icon on the product card or within the product page. 3. The system captures the product's unique ID. 4. The system checks if the product is already in the user's cart. 5. If not, the system adds the product to the cart. If yes, the system updates the quantity.	The product is added to the shopping cart, and the quantity is updated if it's already there.	Added a product to the cart from product page then checked shopping cart, quantity was 1, then I returned to the product page and clicked add to cart again, opened shopping cart and it was now 2.	Pass
TC02	Add to Wishlist Functionality	1. Ensure the product has a unique ID visible to the system. 2. User clicks on the "Add to Wishlist" link on a product card or within the product page. 3. The system retrieves the product's unique ID. 4. The system adds the product to the user's Wishlist in the database.	The product is added to the Wishlist, the interface updates, and the user receives visual confirmation.	Product has been added to the wishlist.	Pass

		5. The system handles any issues, such as the product already being in the Wishlist.			
TC03	Proceed to Checkout Functionality	1. User views the shopping cart and clicks on the “Proceed to checkout” button. 2. The system confirms that there are items in the cart. 3. The system initiates the process to redirect the user to the checkout page. 4. The system ensures all cart information is transferred to the checkout process.	The user is redirected to the checkout page with all items, quantities, and prices correctly listed.	Redirected to the checkout page with the correct products, quantities, and prices.	Pass
TC04	Complete Order Functionality	1. User enters valid shipping information. 2. User clicks on the “Complete order” button. 3. The system validates shipping information and user credentials. 4. The system generates a unique order ID and inputs a new entry in the database. 5. The system sets the order status to ‘Order received’ and redirects the user to the confirmation page.	The user is redirected to an order confirmation page with a unique order ID and summary. The order is saved and linked to the user's account with a	Redirected to the order confirmation page, then checked order on account page. Order details matches with the details inputted in the checkout, Order has a unique ID, and the Order status is set to ‘Order Received’.	Pass

		6. The system updates the user's account page with the new order.	‘Order Received’ status.		
TC05	Check My Order Functionality	1. User clicks on the “Check My Order” button or link. 2. The system verifies the user is logged in and has the necessary permissions. 3. The system retrieves the user’s order history from the database. 4. The system initiates redirection to the account page, focusing on the “Order Section.”	User is redirected to the order section on their account page where they can view their order history.	On Clicking check my order, I have been redirected to the Order section in the account page.	Pass
TC06	Edit Order Functionality	1. User clicks on the “Edit order” link. 2. The system verifies the user’s session. 3. The system retrieves the current state of the user’s cart. 4. The system redirects the user to the shopping cart view. 5. The system displays the contents of the shopping cart, allowing modifications.	User is redirected from the checkout page to the shopping cart page where they can update their cart before returning to checkout.	Upon clicking the ‘edit order’ in the checkout, I was redirected to the shopping cart.	Pass

TC07	Pagination Functionality	1. User clicks on a pagination button (e.g., a number, “Next,” “Previous,” “First Page,” “Last Page”). 2. The system calculates the new page number and range of products to display. 3. The system fetches and displays the new set of products. 4. The system updates the pagination controls to reflect the current page.	The shop page displays a new set of products corresponding to the selected page number. Pagination controls update accordingly.	When clicking next, refreshes the page, the showing label displays the products displayed, and the products are changed as well.	Pass
TC08	Carousel Navigation Functionality	1. User clicks the arrow button (left or right) to navigate through carousel images. 2. The system detects the direction of navigation. 3. The system updates the current image index and applies a transition effect. 4. A new image or content frame is displayed in the carousel. 5. Visual indicators on the carousel update to represent the current image.	The carousel shows the new image or content frame. Visual indicators update to show the current image.	Clicking the arrows changes the images in the carousel.	Pass

TC09	View Product Description	1. User clicks the “Description” option in the bottom menu. 2. The system queries the database for the product’s description using the product ID. 3. The product description is retrieved and loaded into the content area. 4. The text is formatted for display.	The product description tab becomes active, and the detailed product description is displayed within the tab’s content area.	Bottom description is visible.	Pass
TC10	Remove from Wishlist	1. User clicks the “Remove from Wishlist” link on a product in the Wishlist page. 2. The system updates the Wishlist by removing the specified product. 3. The system updates the database accordingly.	The user interface updates to reflect the removal.	Upon clicking ‘remove from wishlist’ button, the page refreshes and the product is gone from the wishlist.	Pass
TC11	View Order Details	1. User clicks on the “Order details” link/button for a specific order. 2. The system fetches and displays the details in a structured modal format.	A modal window displays detailed order information, including products,	On Clicking ‘Order Details’ button, a modal opens, displaying order details, including products, quantity, address details, order ID.	Pass

			shipping address, and order status.		
TC12	Update Account Details	1. User enters new details into the appropriate fields and clicks the “update” button. 2. The system captures and validates the new details. 3. If valid, updates are processed and confirmed. 4. If invalid, the user is notified of the issue.	Upon successful update, the user’s details in the database are updated and updates details in the account page. If unsuccessful, the user receives an appropriate error message.	Inputted different details in First Name, Last name and clicked ‘Update details’, Page refreshed, and details are changed even in the profile overview.	Pass
TC13	Change Password	1. User inputs the current password, new password, and confirms the new password. 2. User clicks on the “Change Password” button. 3. The system authenticates the current password and validates the new password.	If successful, the password is updated, and a confirmation message is displayed. If unsuccessful, the	Upon inputting password details and clicking change password, the page refreshes and displays ‘Password changed successfully’.	Pass

		4. If all checks pass, the system updates the password in the database.	user receives an error message detailing the issue.		
TC14	Edit Address Information	1. User clicks the “Edit” button for a specific address. 2. The system presents a form pre-populated with the existing address. 3. User makes changes and clicks “Save.” 4. The system validates and processes the updated address details.	If successful, the address is update. If unsuccessful, the user is shown an error message.	On Clicking edit, a modal opens with the current details, after inputting the details and clicking save, the page refreshes and the data is changed.	Pass
TC15	Delete Address	1. User clicks the “Delete” button for a specific address. 2. The system presents a confirmation dialog. 3. Once confirmed, the system processes the deletion.	If successful, the address is removed. If unsuccessful, the user receives an error message.	On Clicking delete, the page refreshes and the address is no longer visible.	Pass

TC16	Cancel Order	1. User clicks the “Cancel Order” button for a specific order. 2. The system checks the order's status and eligibility for cancellation. 3. If eligible, the system updates the order status to “cancelled.”	If successful, the order status is updated to “cancelled”. If unsuccessful, the user receives an error message explaining the issue.	After clicking cancel order, the page refreshes and the data is updated. The order is set as ‘Cancelled’.	Pass
TC17	Interacting with UI Elements	1. User interacts with different UI elements such as product categories, brand links, or the search bar. 2. The system captures the event and identifies the selection. 3. The system fetches corresponding data from the database.	The application updates the shop page view with relevant data, displaying the items or information pertinent to the user's selection.	When clicking categories and brands, the page refreshes and shows the products related to the selected brand or category. When using the search bar and searching for a specific product name, the shop page is refreshed, and it shows the products with that name.	Pass

TC18	Person-Icon Button Functionality	1. User clicks on the “Person-Icon” button. 2. The system checks for an active session and user information. 3. Based on the session's status, the system displays the sidebar with appropriate options or the login form.	If a session is active, a sidebar with account navigation options is displayed. If no session is active, a login form appears.	While logged on, the person-icon redirected me to the account page. While logged off, the person-icon opened a side bar, showing the login form and options such as, create account or forgot password.	Pass
TC19	Send Verification Code Functionality	1. User enters their email address and clicks “Send Verification Code.” 2. The system validates the email format and checks its existence in the database. 3. If the email exists, the system sends a verification code to the address.	If the email is in the database, the user is notified that a verification code has been sent. If the email does not exist or an error occurs, an error message is displayed.	Upon entering my email address, the page refreshes and shows me the message: “Your password has been reset, please check your email inbox”. When opening my email inbox, I received an email which included a new password. Then I tried to login using the password provided and it logged me in.	Pass

TC20	Wishlist Access via Gift-Icon	1. User clicks the “Gift-Icon” button in the navbar. 2. The server checks for an active session. 3. If a session is active, the server queries the database for the user's Wishlist items. 4. If no session is active, the system indicates that the user needs to log in.	If logged in, the user is presented with their Wishlist page showing all saved items. If not logged in, the user is prompted to log in.	When logged in, I am redirected to the wishlist section in the account page, when not logged in, I am redirected to the create account page.	Pass
TC21	Contact Form Submission	1. User fills out the contact form with their details and message. 2. User clicks the “Send Message” button. 3. The client-side script validates the input, and if passed, the data is sent to the server. 4. The server performs additional validation and processes the message.	If successful, the user receives a confirmation on the website and an automatic email. If unsuccessful, an error message is displayed.	Upon entering details and clicking on the ‘Sent message’, The page refreshes and a message is shown: ‘Message Sent’. When checking the inbox which should receive email sent from the contact form, a new email is received.	Pass

TC22	Cart Quantity Adjustment	1. User clicks the “plus” or “minus” button to adjust the quantity of an item. 2. The system captures the event and updates the quantity. 3. The system recalculates the total cost and updates the display.	The quantity of the item and the total price are updated accordingly on the shopping cart sidebar and cart page.	When clicking plus, the quantity increased by 1 and price is calculated for the product quantity price and total price. When clicking on the minus, the quantity is reduced by 1 and the price is subtracted from both the product quantity price and total.	Pass
TC23	Login/Signup Form Toggle	1. User clicks on the “Log in” or “Sign up” button, depending on the current form. 2. A JavaScript function toggles the visibility between the login and sign-up forms.	The form on the page changes: if “Log in” is clicked, the sign-up form is hidden and the login form is displayed, and vice versa.	When clicking either ‘log in’ or ‘sign up’ button, the form changes from login to signup or vice versa, depending on the button clicked.	Pass

TC24	Shopping Cart Sidebar Display	1. User clicks on the “Shopping cart” button. 2. The system checks the current state of the sidebar and displays it accordingly.	The sidebar becomes visible on the side of the screen, displaying the contents of the shopping cart.	Clicked the ‘Shopping cart’ button in the navbar, and a side bar opened showing the shopping cart with the products I added.	Pass
TC25	Direct Product Access	1. User clicks the “Product image” or the “title” on a product card. 2. The system retrieves and displays the detailed product page based on the product ID.	The user is presented with the product page, including all relevant details for the selected product.	Clicked on a product on the homepage and I was redirected in a product page which included all the details of the product I had selected.	Pass
TC26	Navigation to Contact Us Page	User clicks the “Contact Us” link on the navbar or footer.	The user is redirected to the contact us page.	Clicked on contact us in the navbar, and I was redirected to the contact us page.	Pass

TC27	Navigation to Policy Pages	User clicks either the “Terms and conditions” link or the “Privacy policy” link in the footer.	The user is redirected to the corresponding policy page.	Clicked on both links and I have been redirected to the corresponding pages.	Pass
TC28	Social Media Redirection	User clicks on one of the social media icons in the footer.	The user is redirected to the corresponding social media platform’s page.	Clicked on all three social media icons and they all redirected me to the corresponding pages.	Pass
TC29	Create Account Access	User clicks on the “Create Account” link in the person button sidebar.	The user is redirected to the create account page.	Redirected to the Create Account Page.	Pass
TC30	Forgot Password Link	User clicks the “Forgot Password?” link.	The user is redirected to the	Redirected to the forgot password page.	Pass

			forgot password page.		
TC31	Sign Out Functionality	User clicks on the “Sign out” button.	The user's session is terminated, and they are redirected to the home page.	Upon clicking the signout button in the account page, I have been redirected to the index page.	Pass
TC32	Home Navigation	User clicks the “Home” button, link, or the “Funzies” logo.	The user is redirected to the homepage.	Redirected to the index/homepage.	Pass
TC33	Log In from Checkout	User clicks on the “Log In” link on checkout page.	The user is redirected to the Login section of the create account page.	Redirected to the login section on the create account page.	Pass

TC34	Shop Navigation	User clicks on the “Shop” link on the navbar.	The user is redirected to the shop page with a display of popular or relevant products.	Redirected to the Shop page.	Pass
TC35	Account Page Navigation	1. User clicks on one of the links like “Profile Overview,” “Addresses,” etc. 2. The corresponding section of the account page is displayed.	The content associated with the clicked link is displayed, and previously visible content is hidden.	Redirected to different section.	Pass
TC36	View Cart Functionality	User clicks the “View Cart” button inside the sidebar.	The user is redirected to the shopping cart page.	Redirected to the shopping cart page.	Pass

TC37	Remove Product from Cart	User clicks the ‘X’ button by the product in the shopping cart.	The product should be removed from the cart.	When clicking the X button next to the product in the shopping cart, the page refreshes, and the product is no longer in the shopping list.	Pass

FUNZIES ADMIN – TEST CASES

ID	Test Case	Steps	Expected Result	Actual Result	Pass/Fail
TC37	Login Authentication	1. Users enter their email addresses and passwords into the login form and click the “submit” button. 2. The server-side script manages the POST request and conducts input validation. 3. The system attempts to authenticate the user against the database credentials. 4. A session is initiated for successful authentication, or an error message is prepared for failure.	If successful, the user is redirected to the admin dashboard. If unsuccessful, an error message is displayed.	Logged in Successfully using an administrator account. Tried to login using a customer account, Got the following error message: ‘Incorrect email or password, try again!’	Pass

TC38	Sign Out Functionality	User clicks the “Sign Out” button.	The user’s session is terminated, and they are redirected to the login page.	I have been logged out and redirected to the login page.	Pass
TC39	Navigation via Management Hyperlinks	User clicks on Brand management hyperlink listed in the sidebar.	The user is redirected to the corresponding management page.	Redirected to Brand Management Page	Pass
TC40	Navigation via Management Hyperlinks	User clicks on Category management hyperlink listed in the sidebar.	The user is redirected to the corresponding management page.	Redirected to Category Management Page	Pass
TC41	Navigation via Management Hyperlinks	User clicks on Order management hyperlink listed in the sidebar.	The user is redirected to the corresponding management page.	Redirected to Order Management Page	Pass

TC42	Navigation via Management Hyperlinks	User clicks on Product management hyperlink listed in the sidebar.	The user is redirected to the corresponding management page.	Redirected to Product Management Page	Pass
TC43	Navigation via Management Hyperlinks	User clicks on Role management hyperlink listed in the sidebar.	The user is redirected to the corresponding management page.	Redirected to Role Management Page	Pass
TC44	Navigation via Management Hyperlinks	User clicks on User management hyperlink listed in the sidebar.	The user is redirected to the corresponding management page.	Redirected to User Management Page	Pass
TC45	Data Editing via Modal	1. User clicks the “Edit” button associated with brand data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is updated on the management page.	Modal is opened, showing editable fields, upon saving, the data changed.	Pass

TC46	Data Editing via Modal	1. User clicks the “Edit” button associated with category data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is updated on the management page.	Modal is opened, showing editable fields, upon saving, the data changed.	Pass
TC47	Data Editing via Modal	1. User clicks the “Edit” button associated with order data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is updated on the management page.	Modal did not show, but the status of the order is visible as a drop down, once the drop-down object is changed and saved, the status order was changed.	Pass
TC48	Data Editing via Modal	1. User clicks the “Edit” button associated with product data. 2. A modal dialog appears with editable inputs for the data.	The modal with editable form fields is displayed, and upon saving, the data	Modal is opened, showing editable fields, upon saving, the data changed.	Pass

		3. User makes changes and clicks “Save” on the modal.	is updated on the management page.		
TC49	Data Editing via Modal	1. User clicks the “Edit” button associated with role data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is updated on the management page.	Modal is opened, showing editable fields, upon saving, the data changed.	Pass
TC50	Data Editing via Modal	1. User clicks the “Edit” button associated with user data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is updated on the management page with a confirmation message or an error message.	Modal is opened, showing editable fields, upon saving, the data changed.	Pass

TC51	Data Deletion	User clicks the “Delete” button associated with product data.	If the deletion is successful, the entry is removed from the page. If unsuccessful, an error message is displayed.	When clicking the delete, the item is no longer visible.	Pass
TC52	Order Cancellation	1. User clicks the “Cancel” button for a specific order. 2. A confirmation prompt appears, and the user confirms the cancellation.	If successful, the order’s status is updated to "cancelled" with a success message. If unsuccessful, an error message is displayed.	No confirmation prompt showed but, when I clicked edit and then changed the drop-down option to cancelled, then I clicked the save button, and the order’s status is changed to cancelled.	Pass
TC53	View Order Summary	User clicks the “View Summary” button for specific order data.	A modal window opens displaying the summary of the order.	No Longer exists due to changes	N/A

TC54	Data Creation via Modal	1. User clicks the “Add new Role” button associated with role data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is created, and it is visible on the management page.	Once I inputted the information and clicked save, the data is shown on page.	Pass
TC55	Data Creation via Modal	1. User clicks the “Add new Brand” button associated with role data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is created, and it is visible on the management page.	Once I inputted the information and clicked save, the data is shown on page.	Pass
TC56	Data Creation via Modal	1. User clicks the “Add new Category” button associated with role data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is created, and it is visible on the management page.	Once I inputted the information and clicked save, the data is shown on page.	Pass

TC57	Data Creation via Modal	1. User clicks the “Add new Product” button associated with role data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is created, and it is visible on the management page.	Once I inputted the information and clicked save, the data is shown on page.	Pass
TC58	Data Creation via Modal	1. User clicks the “Add new Order Status” button associated with role data. 2. A modal dialog appears with editable inputs for the data. 3. User makes changes and clicks “Save” on the modal.	The modal with editable form fields is displayed, and upon saving, the data is created, and it is visible on the management page.	Once I inputted the information and clicked save, the data is shown on page.	Pass