Funzies - Collectibles

Summary

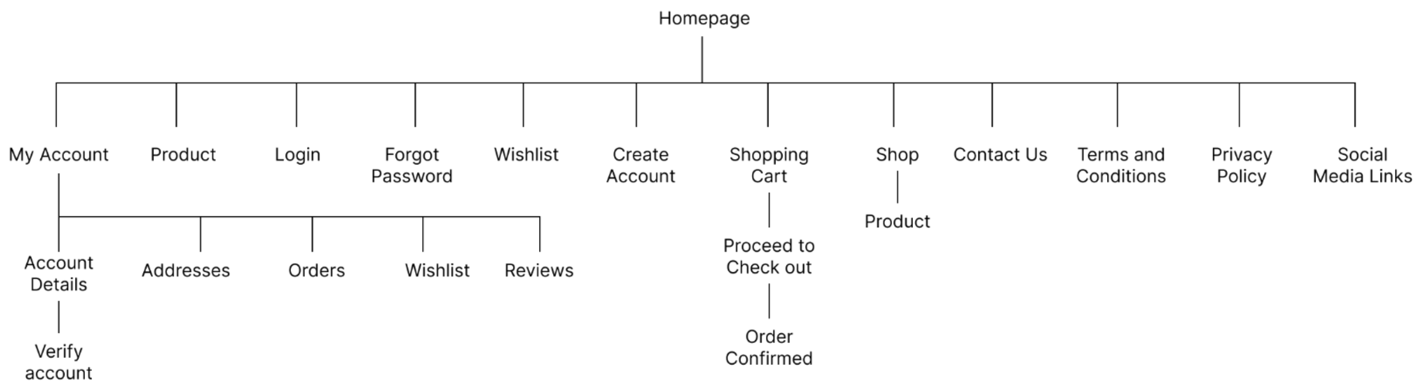
Purpose and Target Audience

Funzies is an e-commerce platform that caters to collectors of various interests and ages, offering a diverse and extensive range of collectibles. This platform is an ideal destination for those seeking items like detailed car models and popular Funko Pop figurines, among other coveted collectibles. It provides an exceptional selection, making it a prime choice for finding unique gifts suitable for collectors and enthusiasts alike.

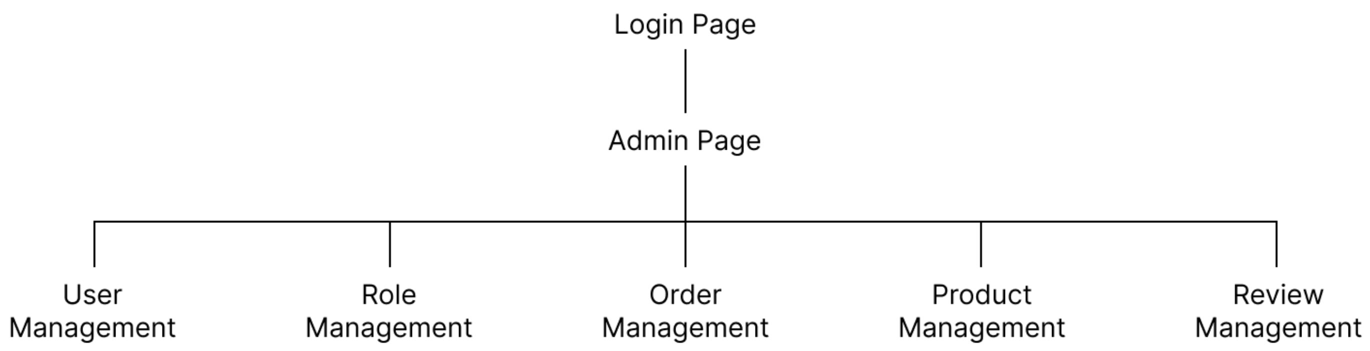
The primary objective of Funzies is to facilitate a seamless and enjoyable shopping experience for collectors. The platform is dedicated to ensuring the availability of unique and extraordinary collectibles, making the process of acquiring these items both convenient and satisfying for customers. Whether it's for augmenting a personal collection or seeking the ideal gift, Funzies offers an accessible and gratifying shopping journey.

Information Architecture and User Experience Design

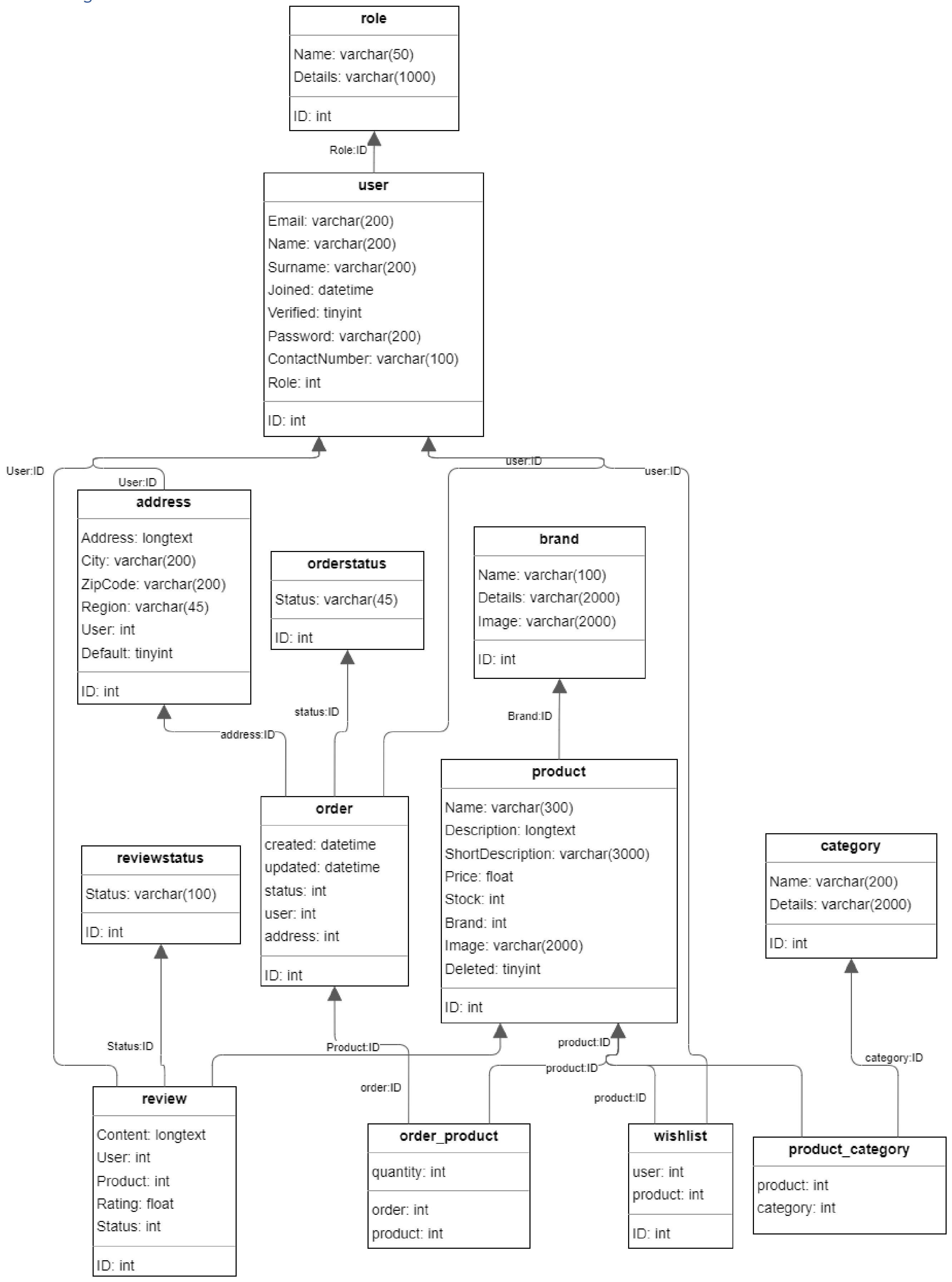
Sitemap – Funzies



Sitemap – Admin Page



UML Diagram



role
Name: varchar(50)
Details: varchar(1000)
ID: int
Role:ID
user
Email: varchar(200)
Name: varchar(200)
Surname: varchar(200)
Joined: datetime
Verified: tinyint
Password: varchar(200)
ContactNumber: varchar(100)
Role: int
ID: int
User:ID
User:ID
user:ID
user:ID
address
Address: longtext
City: varchar(200)
ZipCode: varchar(200)
Region: varchar(45)
User: int
Default: tinyint
ID: int
orderstatus
Status: varchar(45)
ID: int
brand
Name: varchar(100)
Details: varchar(2000)
Image: varchar(2000)
ID: int
status:ID
address:ID
Brand:ID
product
Name: varchar(300)
Description: longtext
ShortDescription: varchar(3000)
Price: float
Stock: int
Brand: int
Image: varchar(2000)
Deleted: tinyint
ID: int
order
created: datetime
updated: datetime
status: int
user: int
address: int
ID: int
reviewstatus
Status: varchar(100)
ID: int
category
Name: varchar(200)
Details: varchar(2000)
ID: int
Status:ID
Product:ID
product:ID
product:ID
order:ID
product:ID
category:ID
review
Content: longtext
User: int
Product: int
Rating: float
Status: int
ID: int
order_product
quantity: int
order: int
product: int
wishlist
user: int
product: int
ID: int
product_category
product: int
category: int

Funzies – Input-Process-Output (IPO) Chart

Input	Process	Output
User clicks on the “Add to Cart” button or icon on a product card or within a product page.	❖ Identify the specific product that the user wants to add to the cart using the product’s unique ID. ❖ Add the product to the user’s shopping cart in the session or database. If the product is already in the cart, update the quantity accordingly.	The item is added to the shopping cart.
User clicks on the “Add to Wishlist” link on a product card or within a product page.	❖ Identify the specific product that the user wishes to add to the Wishlist by retrieving the product’s unique ID. ❖ Add the identified product to the user’s Wishlist in the database. ❖ Manage potential issues such as the product already existing in the Wishlist.	❖ The product is successfully added to the user’s Wishlist. ❖ The user interface updates the product link from “Add to Wishlist” to “Remove from Wishlist). ❖ The user receives visual confirmation that the action has been successful.
User clicks on the “Proceed to checkout” button on the view shopping cart page.	❖ Confirm that there are items in the cart to proceed with checkout. ❖ Initiate the process to redirect the user to the checkout page. ❖ Ensure all cart information, such as items, quantities, and prices, are transferred to the checkout process.	❖ Redirect the user to the checkout page. ❖ The checkout page is pre-populated with all items from the cart, along with their

		quantities and correct pricing.
User clicks on the “Complete order” button on the checkout page.	❖ Validate shipping information entered by the user. ❖ Authenticate user credentials and confirm session validity. ❖ Generate a unique order ID for the new transaction. ❖ Input new entry in the database which includes the Product details (IDs, names, quantities, prices), User account association (linking the order to the user’s account), Shipping information. ❖ Set the initial status of the order to “Processing” within the database. ❖ Redirect the user to the order confirmation page which includes the unique order ID and summary. ❖ Update the user’s account page to include the new order in the “Orders Section.”	❖ The user is redirected to an order confirmation page that displays their unique order ID along with a summary of the order. ❖ The order is saved in the database with all relevant details and linked to the user’s account. ❖ The order status is set to “Processing” in the system, and it becomes visible in the “Orders Section” on the user’s account page for future reference.
User clicks on a “Check My Order” button or link on the order confirmation page.	❖ Verify that the user is logged in and has the necessary permissions to view their orders. ❖ Retrieve the user’s order history information from the database based on their user ID or session.	User is redirected to the order section in the account page.

	❖ Initiate redirection to the account page specifically focusing on the “Order Section.”	
User clicks on the “Edit order” link on the checkout page	❖ Verify the user’s session to ensure continuity of the shopping experience. ❖ Retrieve the current state of the user’s cart from the session or database. ❖ Initiate the redirection process to the shopping cart view. ❖ Prepare and display the contents of the shopping cart, allowing the user to see the items they have added.	❖ User is redirected from the checkout page to the shopping cart page. ❖ User can update their cart as needed before proceeding back to the checkout.
Users click on a pagination buttons (e.g., a number, “Next,” “Previous,” “First Page,” “Last Page”).	❖ Determine the new page number based on the button clicked. ❖ Calculate the range of products to be displayed. ❖ Execute a database query to fetch the product information for the corresponding product card. ❖ Retrieve product data from the database including images, descriptions, prices, etc., for the new set of product cards to be displayed. ❖ Clear or replace the current list of products displayed with the new page’s products. ❖ Update the state of the pagination buttons to reflect the current page button on the page.	❖ The shop page displays a new set of products corresponding to the selected page number. ❖ Pagination controls reflect the current page status, with the correct page number highlighted or disabled buttons as appropriate.
Users click on the arrow button (left or right) to navigate through the carousel images.	❖ Detect which button has been clicked by the user (left arrow, right arrow, or specific block button).	❖ The carousel shows the new image or content frame.

	❖ If an arrow button is clicked, determine the direction of navigation. ❖ For arrow buttons, increment or decrement the current image index to move to the next or previous image in the sequence. ❖ Apply a transition effect for moving between images. ❖ New image in the carousel's display area, replacing the previous one.	❖ Visual indicators on the carousel update to represent the current image being displayed.
User clicks the "Description" option in the bottom menu of the product page.	❖ Based on the product ID, the system queries the database for the product's description. ❖ Retrieve the product description from the database, ensuring it corresponds to the correct product ID. ❖ Load the retrieved product description into the tab's content area. ❖ Format the product description text for display.	❖ The product description tab on the product page becomes active, indicating to the user that it is selected. ❖ The detailed product description is displayed within the tab's content area.
User selection of the "Customer Reviews" option in the bottom menu of the product page.	❖ Initiate a query to the product database using the product's unique identifier to fetch review data. ❖ Apply filters to select only approved reviews from the database. ❖ Format the data for display, such as sorting by date, helpfulness, or rating. ❖ Retrieve and compile a list of approved reviews from the server's database.	A display on the product page under the "Customer Reviews" section that lists all approved reviews related to the product.

	❖ Organize the retrieved reviews into a user-friendly format, including reviewer name, date, rating, and comment.	
❖ User's text and/or rating input into the review form on the product page. ❖ User's action of clicking the "Submit" button in the review section.	❖ Validate the input to ensure it meets the platform's standards for reviews. ❖ The review is stored in the database with the status set as "processing." ❖ Queue the submitted review for administrative approval. ❖ Provide immediate feedback to the user that the review has been submitted and is pending approval.	A message to the user confirming the submission of their review and indicating that it is pending approval.
User clicks the "Remove from Wishlist" link found on the product card in the Wishlist page	❖ The system updates the user's Wishlist by removing the specified product entry. ❖ The system updates the database to reflect the change in the user's Wishlist.	❖ An updated user interface where the removed product no longer appears in the Wishlist. ❖ A message is displayed indicating that the product has been successfully removed from the Wishlist.
User clicks on "Order details" link/button found in the account overview and orders section in the Account Page	❖ Retrieve the list of orders associated with the user's account. ❖ When a specific order is selected, trigger a modal.	❖ A modal window displayed on the user's screen with the following details: ❖ Title of the Modal: Order ID.

	❖ Fetch order details from the database, including products, shipping address, order ID, and user’s name. ❖ Organize the fetched information into a structured layout for the modal. ❖ Set the order ID in the title of the modal. ❖ Display the user’s name, address, and ordered products within the body of the modal.	❖ Body Content: ❖ User’s full name. ❖ Detailed list of products ordered. ❖ Shipping address associated with the order. ❖ Any other relevant order information (date and status).
User enters their new account details into the appropriate fields and clicks “update” button on the account details section on the account page.	❖ The system captures all the new details entered by the user. ❖ The system validates the new details. ❖ If validation passes, the system sends the updated details to the server for processing. ❖ The server updates the relevant user details in the database. ❖ The system generates a confirmation of the update process. ❖ If validation fails or the server encounters an issue, the system captures this and prepares to notify the user.	Successful Update: ❖ The user’s details in the database are updated. ❖ The account details page reflects the new changes. ❖ A confirmation message is displayed to the user, indicating the update was successful. Unsuccessful Update: ❖ The user is presented with an error message. ❖ The original details remain unchanged.

User inputs their new email address in the provided field and clicks the “Update Email” button on the account page.	❖ The system validates the format of the new email address. ❖ The system sends an email to the previous email address, containing a verification link. ❖ The system sets a time frame during which the verification must be completed. ❖ If the user clicks the verification link or inputs the code from the old email, the system finalizes the email update. ❖ If the user does not verify the change within the set time frame, or if the new email is already in use or invalid, the system retains the old email address and provides an error message.	Successful Email Update: ❖ The new email address is confirmed and fully updated in the user’s account details in the database. ❖ A confirmation message is sent to the new email address, and the user interface reflects the change. Unsuccessful Email Update: ❖ The user is informed that verification is pending and reminded to check their old email for the verification link.
User clicks on the “Change Password” button within the account details section after inputting the following: ❖ Current password ❖ New password	❖ The system compares the entered current password with the password stored in the database for authentication. ❖ The system checks if the new password and confirm new password inputs match each other. ❖ The system validates the new password requirements.	Successful Update: ❖ The database confirms the password has been updated. ❖ A confirmation message is displayed to the user, indicating

❖ Confirm password.	❖ If all checks passed, the system proceeds to update the password in the database. ❖ If there is any mismatch or validation failure, the process is halted, and errors are captured.	that their password has been successfully changed. Unsuccessful Update: ❖ The user receives an error message.
The user clicks an “Edit” button associated with a specific address in the addresses section on the account page.	❖ The system opens a modal window containing a form. This form would be pre-populated with the existing information for the selected address. ❖ The user makes changes to the address information in the provided form fields. ❖ Input validation is performed to ensure the new address data meets the system’s requirements. ❖ Upon the user clicking “Save,” the system captures the updated address details. ❖ The system sends the updated information to the server for processing. ❖ The server receives the updated address information, authenticates the user’s session, and updates the address details in the database.	❖ The database confirms the address has been updated. ❖ The modal window closes, and the user interface refreshes the address list to show the updated address details. ❖ A confirmation message is displayed to the user, indicating the address has been successfully updated. ❖ If there is an issue, the user is shown an error message.
The user clicks a “Delete” button associated with a specific address in the addresses section of the account page.	❖ A confirmation dialog is presented to the user to prevent accidental deletion. ❖ Once confirmed, a request is sent to the server to delete the address. ❖ The server processes the request, verifies the user’s permission, and deletes the address from the database based on the ID.	❖ The user is notified that the address has been successfully deleted. ❖ The address is removed from the list

		of addresses on the account page. ❖ If the deletion cannot be completed the user receives an error message.
The user initiates an order cancellation by clicking a “Cancel Order” button associated with a specific order in the orders section of the account page.	❖ The server checks the status of the order using the order ID. ❖ It is verified whether the order is in a state that allows cancellation (e.g., not shipped or delivered). ❖ If the order is eligible for cancellation: ❖ The server updates the order record in the database, changing the order status to “cancelled.” ❖ Any further actions related to the order, such as edit or delete, are disabled to reflect the cancellation.	❖ If the order status is successfully changed to “cancelled,” the user receives a confirmation message. ❖ If the order cannot be cancelled because it has been shipped or delivered, the user receives an error message explaining why the cancellation is not possible. ❖ For a successful cancellation, the order details are updated to reflect the new “cancelled” status, and interactive buttons like “edit” or “delete” are disabled or removed.

		❖ If there is an error during the process: an error message is displayed, and no actions are taken.
The user clicks an “Edit” button or link associated with a specific review on the review section in the account page.	❖ The user makes changes to the review content within the form. ❖ When the user clicks “Save,” the updated review is sent to the server. ❖ The server verifies that the user is authorized to edit the review. ❖ The server updates the review record in the database with the updated content and sets its status to “processing.” ❖ The review is temporarily hidden from the product page while it is in “processing” status, pending approval or review.	❖ After saving, the user receives a confirmation message indicating that the review has been updated and is under processing. ❖ The account reviews section is refreshed to show the updated review. ❖ On the product page, the review is not visible. ❖ If unsuccessful: an error message is displayed.
User clicks a “Delete” button or link associated with a specific review on the review section of the account page.	❖ A confirmation popup appears asking the user to confirm the deletion of the review. ❖ If the user confirms: ❖ Upon confirmation, a request is sent to the server with the review ID. ❖ The server receives the request and authenticates the user to ensure they have permission to delete the review.	❖ The user is informed of the successful deletion by the removal of the review and a confirmation message. ❖ The review is no longer visible on the

	❖ The server then searches the database for the review using the provided ID. ❖ If found, the server updates the review’s status to “deleted.” ❖ The user is notified that the review has been deleted. ❖ If the user cancels, the process stops here, and no further action is taken.	product page or the account page. ❖ If an error occurs, an error message is displayed.
User interaction through UI elements: ❖ Product categories options ❖ Brand name links on product cards ❖ Brand images in “shop by brand” menu ❖ “More products” button on home page and product pages ❖ Category buttons in “shop by categories” menu ❖ Search term input and submit button.	❖ Server request is triggered by click events or search submission. ❖ Identify selection (category, brand, new arrivals, best sellers, or search term). ❖ Fetch corresponding data from the database.	❖ Application prepares the shop page view with appropriate data. ❖ The shop page is updated and displayed to the user with the relevant item list.

❖ Category options in the burger menu ❖ Drop down options in the shop page		
User clicks on the "Person-Icon" button in the navbar	❖ The PHP script checks and verifies whether there is a session in progress by checking session variables. ❖ If a session is active, it gathers the necessary information about the user to determine which account page options to display. ❖ If no session is active, it prepares to respond with a prompt to display the login form. ❖ The PHP script sends a response back to the Bootstrap function based on the session check. ❖ If the session is active, Bootstrap elements display the sidebar with user account options. ❖ If the session is not active, Bootstrap triggers the display of the sidebar with the login form.	❖ If a session is active, the output will be a sidebar displaying various options for the user to navigate through various parts of their account page, such as "Profile Overview," "Orders," "Wishlist," and a "Logout" option. ❖ If no session is active, a login form will be displayed for the user to sign in.
❖ user clicks on a "Person" icon or button that opens a sidebar with login fields or find the	❖ It retrieves the username and password input by the user. ❖ If the "Remember Me" is checked, the system will set a persistent cookie to save the login credentials.	If log in was Successful: ❖ The user is redirected to their user profile page or dashboard, or to the page they were

Login form in the create account page. ❖ The user enters their username and password into the login fields. ❖ The user may also click a "Remember Me" checkbox to indicate they want their login details to be remembered on the device.	❖ The system checks the database to verify the credentials. ❖ If credentials are correct, the system logs the user in.	attempting to access before being prompted to log in. ❖ The UI updates to reflect the user's logged-in status, such as displaying a logout button, account links in the sidebar and providing access to user-specific features such as writing, editing, and deleting reviews. If log in is unsuccessful: ❖ The user is presented with an error message, such as "Invalid username or password." ❖ The UI remains unchanged.
The user clicks the "Send Verification Code" button after entering their email address.	❖ The email format is validated using a regular expression to ensure it meets the standard email format. ❖ The validated email is sent to the server. ❖ The server queries the database to confirm whether the email address is associated with an existing account.	❖ If the email exists in the database, the user is notified that a verification code has been sent to their email and a verification code is

	❖ If the email exists, the server generates a verification code, or a password reset token and is given an expiration time for the code/token. ❖ A system-generated email containing the verification code is sent to the user's email address.	delivered to the user's email address. ❖ If the email does not exist, an error message will be displayed to let the user know that the email address is not recognized. ❖ If any errors occur during the process the user is informed of the failure to send the verification code.
User clicks the "Gift-Icon" button in the navbar	❖ The server to checks for an active session. ❖ The PHP script on the server verifies the presence of an active session by checking session. ❖ If a session is confirmed, the script then queries the database for the user's saved items in their Wishlist. ❖ If no session is found, the script prepares a response indicating that the user is not logged in.	❖ If the user is logged in, they are presented with their Wishlist page showing all saved items. ❖ If the user is not logged in, they are prompted to log in to view their Wishlist.
❖ User fills out the contact form with details such as name, email, subject, and message.	❖ The client-side script validates the input using JavaScript Regex to ensure all fields are filled correctly.	If unsuccessful: ❖ The user is informed with an error message detailing the problem.

❖ User clicks the “Send Message” button on the Contact us page	❖ If client-side validation is passed, the form data is packaged into an HTTP request and sent to the server. ❖ The server receives the request and performs its own validation check. If the data is valid the message and the user’s contact information are used to be sent through the mailing system to the funzie’s mailbox. ❖ The system generates an automatic reply to the user’s email address confirming receipt of the message.	If Successful: ❖ The user is presented with a confirmation message on the website indicating that their message has been sent successfully and will receive an automatic email confirming that their message has been received and will be addressed.
The “plus” and “minus” buttons found in the shopping cart sidebar and view chart page	❖ The system captures the button click event and identifies which item it pertains to. ❖ It then increments or decrements the quantity of that item in the cart. ❖ The system recalculates the total cost. ❖ The updated cart, including the new quantities and total, is displayed to the user. ❖ The system ensures the session database reflects the updated cart state.	❖ The displayed quantity of the item is updated. ❖ The shopping cart sidebar and the cart page reflect the new quantity. ❖ The updated total price is displayed.
User clicks on the “Log in” or “Sign up” button, depending on which form is currently active/inactive on the page.	❖ The browser detects the click event on the “Log in” or “Sign up” button. ❖ A JavaScript function is called when the button is clicked. This function determines which form is currently visible and which one needs to be displayed.	The form on the “Create Account” page changes. If the “Log in” button was clicked, the sign-up form is hidden, and the login form is displayed. Conversely, if the “Sign

	❖ The JavaScript function will hide one form and show the other. ❖ The state of the UI is updated to reflect which form is active.	up" button was clicked, the login form is hidden, and the sign-up form is displayed.
User clicks on the "Shopping cart" button	❖ Bootstrap checks the current state of the sidebar (whether it is open or not). ❖ Based on the state, display change to the sidebar into view.	The sidebar becomes visible on the side of the screen, displaying the contents of the shopping cart.
User clicks the "Product image" or/and the "title" on the product card	❖ The product image and title are associated with a particular product ID, which is sent to the server-side application. ❖ The server-side application receives the request and extracts the product ID from the product page URL. ❖ The application then queries the database or data source for the product details corresponding to the provided ID. ❖ The product page template is populated with the retrieved product data, creating a complete and detailed page for the product.	The user is presented with the product page that includes all relevant data for the product they selected.
User clicks the "Contact Us" link on the navbar and footer	❖ The user's browser interprets the click as a request for the contact us page	❖ The user is redirected to the contact us page
User clicks either the "Terms and conditions"	❖ It identifies the URL associated with the clicked icon.	The user is redirected to the corresponding page.

link or the “Privacy policy” link in the footer	❖ The browser then sends a request to that URL.	
User clicks on one of the social media icons in the website footer (e.g., Facebook, Twitter, Instagram).	❖ It identifies the URL associated with the clicked icon, which is the target social media platform’s homepage or the website’s specific page on that platform. ❖ The browser then sends a request to that URL.	The user is redirected to the corresponding social media platform’s page.
User clicks on the “Create Account” link in the person button sidebar	The browser detects the click event on the “Create Account” link.	The user is redirected to the create account page.
User clicks the “Forgot Password?” link on a login form.	The browser detects the click event on the “Forgot Password?” link.	The user is redirected to the forgot password page.
User clicks on the “Sign out” button in the account page.	The server-side script is triggered by the click which then proceeds to terminate the user’s current session.	The user is redirected to the home page.
❖ User clicks the “Home” button on the order confirmed page. ❖ User clicks the “Home” link or the “Funzies” logo located in the footer or navbar.	❖ The user’s browser interprets the click as a request for the home page	❖ The user is redirected to the homepage

User clicks on the "Log In" link on the checkout page	The browser detects the click event on the "Log In" link. The browser reads the link attribute of the link to determine the destination URL.	The user is redirected to the Login section of the create account page.
User clicks on the "Shop" link on the navbar	❖ The click event triggers a request to the server. ❖ The application processes the request to identify the most popular products. ❖ The application prepares the shop page view and incorporate the data.	❖ The user is redirected to the shop page
The user fills in the required details on the sign-up form such as name, email address, password, etc., and clicks the "Sign Up" button	❖ When the user submits the form, before sending the data to the server, the client-side script uses regex to validate the format of the data. ❖ Email regex checks if the user's input matches the pattern of a valid email address. ❖ Password regex verifies the password meets certain criteria. ❖ If the regex validation is successful, the data is sent to the server via an HTTP POST request. ❖ If all validations pass, the server proceeds with creating the user account. ❖ The server sends a verification email to the provided email address.	❖ The user receives immediate feedback if the data entered does not match the required patterns, in the form of error messages next to the respective form fields. If Unsuccessful: ❖ The user will receive an error message. If successful: ❖ The user receives a verification email to proceed with account activation.

When a user clicks on one of the links such as “Profile Overview,” “Account Details,” “Addresses,” “Orders,” “Wishlist,” or “Reviews” on the account page.	Bootstrap-powered JavaScript function captures the click and looks for the content which contains the triggered ID. The corresponding div element in the main content area is targeted. The collapse attribute is triggered, and the corresponding content is called and displayed.	The visual output of these processes is the collapsed in the user interface. When a user clicks on a list-group-item, the associated section will be displayed, while the visible section will be hidden.
User clicks the “View Cart” button inside the sidebar	The application responds to this event by navigating to the shopping cart page.	User is redirected to the Shopping cart page.

Admin Dashboard – Input-Process-Output (IPO) Chart

Input	Process	Output

Users enter their email addresses and passwords into the login form and then click the “submit” button.	❖ The server-side PHP script manages the POST request when the user submits the form. ❖ Input validation is conducted to ensure that the email and password meet the required format and security checks, mitigating issues like SQL injection. ❖ The system attempts to authenticate the user by verifying the provided email and password against the stored credentials in the database. ❖ If the credentials are correct, the system initiates a user session and stores the necessary session variables. ❖ In case the credentials are incorrect, the system prepares to send back an error message to inform the user.	If successful: ❖ The user is redirected to the admin dashboard page. If unsuccessful: ❖ The user is presented with an error message that describes the issue encountered.
User clicks the “Sign Out” button.	The User’s session is terminated.	The user is redirected to the login page.
User clicks on one of the management hyperlinks listed in the sidebar to navigate to the respective management page.	The server retrieves the content of the target management page.	The user is redirected to the corresponding page.
User clicks the “Edit” button associated with the specific data	The ‘Edit’ button triggers a JavaScript function or event listener that opens a modal	The modal with the editable form fields will be displayed to the user.

found on the management pages.	dialog on the page, displaying editable inputs with the information to update.	
The user clicks "Save" on the modal.	❖ The form submission is managed directly by PHP on the server-side via a POST request. ❖ The server receives the edited data and processes it, updating the corresponding details in the database.	If the update is successful: ❖ A confirmation message is displayed to the user. ❖ The order details on the management page are updated. If the update is unsuccessful: ❖ An error message is shown to the user. ❖ The details on the management page remain unchanged.
User clicks the "Delete" button associated with the specific data found on the management pages.	❖ The server-side script processes the delete request. ❖ The script executes a SQL DELETE command to remove the entry from the database. ❖ The script checks the success of the deletion and prepares the response.	If the deletion is successful: ❖ A confirmation message is displayed to the user. ❖ The entry is removed from the management page. If the deletion is unsuccessful: ❖ An error message is displayed explains why the deletion could not be performed.
User clicks the "Cancel" button associated with the	❖ The system identifies the specific order for cancellation using the order ID.	If the cancellation is successful:

specific data found on the Order management page.	❖ A confirmation prompt is displayed to the user to prevent accidental cancellations. ❖ If the user confirms the cancellation, the server-side script processes the request. ❖ The script checks the order's status to ensure it can be cancelled (e.g., not already shipped or previously cancelled). ❖ The script updates the order status to 'cancelled' in the database.	❖ A success message is displayed to the user, and the order's status is visually updated on the page. If the cancellation is unsuccessful: ❖ An error message is displayed to the user, providing an explanation for why the order could not be cancelled.
User clicks the "View Summary" button associated with the specific data found on the Order management page.	❖ The system retrieves the order ID. ❖ A request is sent to the server-side script to fetch the details of the selected order. ❖ The server-side script executes a database query to retrieve the order details, including order ID, product details, price, order status, customer ID, and customer address. ❖ The retrieved data is formatted for display.	A modal window opens on the current page, displaying the summary of the order.
User clicks the "Approve" button associated with the specific data found on the Review management page.	❖ The system identifies the review ID associated with the button. ❖ A server-side request is initiated to manage the approval action.	If the approval is successful: ❖ The table is updated to reflect that the review has been approved. ❖ The approved review becomes visible on the

	❖ The server-side script validates the review's status to ensure it can be approved. ❖ Upon validation, the script updates the review's status to 'approved' in the database. ❖ The review on the product page becomes visible. ❖ The system confirms the update and prepares to notify the user interface of the change.	product page for other customers to see. If the approval is unsuccessful: ❖ An error message is displayed, and the review's status remains unchanged.
User clicks the "Decline" button associated with the specific data found on the Review management page.	❖ The system retrieves the ID of the review to be declined. ❖ The script verifies the review's status to ensure it can be declined. ❖ The review status in the database is updated to 'declined.' ❖ A notification process is triggered to inform the review's author. ❖ The system logs this action and updates the review records accordingly.	If the decline is successful: ❖ The status of the review is updated to "declined." ❖ The author of the review is sent an email instructing them to resubmit their review. If the decline is unsuccessful: ❖ An error message is displayed to the user attempting to decline. ❖ The review status remains unchanged, and no message is sent to the author.