Perfect Present

models

app/models/user.rb

- Included the default devise modules
- Configured devise to use database authentication, registration, password recovery, rememberable, and validations
- Defined a relationship between User and Wishlist models where a user has many wishlists
- Set up the dependency between the User and Wishlist models to destroy all associated wishlists when a user is destroyed

app/models/wishlist.rb

- Created a class called Wishlist which inherits from ApplicationRecord.
- Defined a relationship between the Wishlist model and the User model using belongs_to :user.
- Added a validation using validates_uniqueness_of to ensure that the combination of api_id and user_id is unique for each Wishlist record.

Perfect Present

controllers
app/controllers/application_controller.rb

- Authenticated the user before executing any action by using before_action :authenticate_user!.
- Defined a method default_url_options in the ApplicationController class that returns a hash containing the key-value pair for host.
- The value of host is either set to the environment variable DOMAIN or "www.perfect-present.me" if DOMAIN is not set.
- This method is inherited by other controllers that inherit from ApplicationController.

app/controllers/pages_controller.rb

- Created a PagesController class that inherits from the ApplicationController class.
- Declared a before_action callback method named "request_module".
- Declared a skip_before_action method that skips the "authenticate_user!" method for the "home" action.
- Defined a "home" method that does not take any parameters.
- Defined a "category" method that takes in two parameters: "hobby" and "age_range".
- Created a hash called "hobbies_category_match" that maps a hobby to an array of categories that match that hobby.
- Created a hash called "ages_category_match" that maps an age range to an array of categories that match that age range.
- Assigned the value of the "hobby" parameter to an instance variable called "@hobbies".
- Assigned the value of the "age_range" parameter to an instance variable called "@age_range".
- Looked up the matching categories for the hobby and age range from the respective hash tables and assigned them to instance variables called "@hobbies_category_match" and "@ages_category_match".
- Rendered the "category" view template, passing in the "@hobbies_category_match" and "@ages_category_match" instance variables.

Perfect Present

controllers

app/controllers/presents_controller.rb

- In the PresentsController class, there's a before_action hook that calls the request_module method before the index and show actions are executed.
- The index action sets the @price_range instance variable to the :price_range parameter passed in the request. It then calls the request_api method with the :category_id parameter and a nil value for the item_id.
- Depending on the value of @price_range, the action selects a subset of the products returned by the API and assigns it to the @products instance variable. If no @price_range value is set, it calls request_api again and assigns the full product list to @products.
- The show action calls request_api with a nil value for the category_id parameter and the :id parameter passed in the request as the item_id. It assigns the product with the matching ID to the @product instance variable.
- The request_api method takes two optional parameters: category_id and item_id. If item_id is nil and category_id is not nil, it builds a URL to the MercadoLibre API to search for products in the specified category. If item_id is not nil, it builds a URL to the API to retrieve information about a single product. It then sends a request to the API using the open-uri library, parses the response JSON, and returns the resulting object.
- Finally, there's a commented-out line of code that would randomly select 3 products from the @products array if uncommented.

app/controllers/wishlists_controller.rb

- Set up a controller for the Wishlists model in Ruby on Rails.
- Define an action for the index page, which retrieves all Wishlist items and filters them to show only those belonging to the current user.
- Call an external API to retrieve details for each Wishlist item that belongs to the current user.
- Define an action for creating a new Wishlist item, which sets the current user as the owner and the API ID as the item to be added.
- Define an action for deleting a Wishlist item, which retrieves the item to be deleted based on its API ID and user ID.
- Define a private method to permit the API ID parameter to be passed in through the form.
- Define a private method to retrieve the Wishlist item to be deleted based on its API ID and user ID.

Perfect Present

views

app/views/pages/category.html.erb

- Initialized variables @categories, @icons, and @count using values from @categories_array.
- Loop through each category in @categories.
- If @count is a multiple of 5, create a new row.
- Create a new card div with the class col col-lg-2 mt-3.
- Inside the card div, create a link with the category name that goes to the presents_path with the category ID and price range as parameters. The link has the class link_style.
- Find the icon associated with the category by looking for the icon in @icons where the first element matches the category ID.
- Inside the card div, create an i element with the class from the found icon and a size of fa-5x.
- Increment @count by 1.
- If @count is a multiple of 5, close the row div.

app/views/shared/_footer.html.erb

- Created a footer section with two divs - "footer-phrase" and "footer-style".
- Inside the "footer-phrase" div, commented out an image tag for an avatar.
- Added an h2 tag with an ID of "footer-text" that displays the text "Created with love by the Perfect Present Team".
- Inside the "footer-style" div, created another div with a class of "follow".
- Added a span tag with a class of "text-muted" and an h4 tag with a class of "h4-footer" and text "Follow Us".
- Created a div with a class of "border-bottom" for styling purposes.
- Added a div with a class of "footer-links social-media" to contain the social media icons.
- Added three anchor tags with href links to Instagram, Facebook, and Twitter.
- Added an image tag inside each anchor tag for each social media platform, with class names of "instagram", "facebook", and "twitter", and image sources that display the respective social media icons.
- Set the width and height of each image tag to 50 pixels.

app/views/pages/home.html.erb

- Created a banner section for the "Perfect Present" page using HTML and ERB syntax.
- Displayed an image using the <img> tag and the Cloudinary service to upload and store the image.
- Commented out a header and a paragraph that were not used.
- Used the Bootstrap class "img-fluid" to make the image responsive and fit the screen size.
- Set the "d-block" class to display the image as a block element, removing any space around it.
- Added a margin-bottom class to give some space between the banner and the next element in the page.

Perfect Present

views

app/views/presents/index.html.erb

- Iterated over an array of products with .each method, assigning each product to a block variable product
- Created a div element with class col mx-auto to hold each product card
- Created a card element with class card h-auto w-auto to display product information
- Included an image tag with source set to the thumbnail attribute of the product object, and alt text left empty
- Included a h5 element with class card-title to display the title attribute of the product object
- Included a p element with class card-text fs-4 to display the price attribute of the product object
- Included a button with text "Check Me Out!" using button_to method, which sends a GET request to the present_path route with the product's id as a parameter, and with class btn btn-primary for styling

app/views/presents/show.html.erb

- Added metadata for SEO optimization using the content_for method.
- Created a div container with a class of card for displaying the product details.
- Set the product title using the @product["title"] instance variable.
- Displayed the product price using the @product["price"] instance variable.
- Displayed the product description using the @product["description"] instance variable.
- Set the product image source using the @product["pictures"][0]["secure_url"] instance variable.
- Created a "Purchase" button that links to the product permalink using the link_to method.
- Created a "Wishlist Me!" button that sends a POST request to the wishlists_path with the api_id parameter set to the product id using the button_to method.

app/views/shared/_homepage_form.html.erb

- Added an image to the container to display the Perfect Present logo.
- Added a paragraph to welcome the user and instruct them to fill out the form below to find the perfect present for their loved one.
- Created a form with action set to "/pages" using the GET method.
- Added select tags for "price_range", "hobby", and "age_range" using options for select and required attributes.
- Added a submit button with text "Submit" and class "btn btn-light button".
- Used form_with with url set to pages_path, local set to true and method set to get to create the form

Perfect Present

views

app/views/shared/_navbar.html.erb

- Created a navbar with the class "navbar navbar-expand-sm navbar-light navbar-lewagon" and a height of 91 pixels.
- Added a container-fluid to center the navbar.
- Created a link to the home page with the navbar-brand class and an image logo.
- Added a button to toggle the navbar collapse with the class "navbar-toggler" and an icon.
- Created a collapse navbar with the class "collapse navbar-collapse" and an unordered list with the class "navbar-nav me-auto".
- Added an if statement to check if a user is signed in.
- Added a div element with the class "border".
- Added a list item with the class "nav-item margin-zero active" and a link to the home page.
- Added another div element with the class "border".
- Added a list item with the class "nav-item dropdown" containing an image tag with the user's avatar and a dropdown menu with links to the wishlist, a logout button and an option to perform an action.
- Added an else statement to show a login link if a user is not signed in.

app/views/shared/_team.html.erb

- Created a flex-container with the class "flex-container container-fluid container-color-3" to display the team section.
- Created a container with the class "container container-team" to center the team section.
- Added a heading with the text "MEET THE TEAM   " and the id "about-subtitle".
- Created four cards with the class "card card-team" to display the team members.
- Added a div element with the class "border border-team" to each card to add a border around the team member's name.
- Added a heading with the class "team-h2" and each team member's name to the div element inside each card.

Perfect Present

views

app/views/shared/_testimonial.html.erb

- Created a flex-container with the class "flex-container container-fluid container-color" to display the idea behind the Perfect Present.
- Created a div element with the class "flex-item" to add a heading with the text "THE IDEA BEHIND" and the id "about-title".
- Created another div element with the class "flex-item mt-5" and id "testimonial-p" to add a paragraph with the text describing the Perfect Present's purpose and how it helps users find the perfect gift.
- Added a container with the class "container mx-auto" to center the testimonials section.
- Added a heading with the text "WORDS FROM OUR CLIENTS" and the id "about-subtitle" to the container.
- Created a container with the id "carousel-testimonials" to display the testimonials section.
- Added a carousel with the id "carouselExampleFade" and the class "carousel slide carousel-fade" to display the testimonials in a sliding format.
- Added two carousel items with the class "carousel-item" to display two sets of testimonials.
- Added a div element with the class "d-flex justify-content-center mx-auto" to center the testimonials' images.
- Added three image tags with the class "d-block mx-3 cards-size" to display the testimonials' images.

app/helpers/meta_tags_helper.rb

- Created a module called MetaTagsHelper to define methods for setting meta tags.
- Created a method called meta_title to check if there is a meta title set in the content_for helper. If true, it returns the value of the meta title. If not, it returns the value of the DEFAULT_META hash with the key "meta_title".
- Created a method called meta_description to check if there is a meta description set in the content_for helper. If true, it returns the value of the meta description. If not, it returns the value of the DEFAULT_META hash with the key "meta_description".
- Created a method called meta_image to check if there is a meta image set in the content_for helper. If true, it returns the value of the meta image. If not, it returns the value of the DEFAULT_META hash with the key "meta_image".
- Added a check to see if the meta image starts with "http". If true, it returns the meta image. If false, it returns the image URL using the image_url helper method. This is to allow the use of both an asset or a URL as the meta image.

Perfect Present

views

app/views/layouts/application.html.erb

- The page title, favicon, and meta tags are defined in the head section of the layout
- The layout loads the main stylesheet and JavaScript files for the application using the stylesheet_link_tag and javascript_pack_tag helper methods
- The layout also sets up some meta tags for Facebook and Twitter sharing using the meta_title, meta_description, and meta_image helper methods from the MetaTagsHelper module
- The yield statement renders the content for each page that uses this layout, which will be inserted into the main section of the layout
- The layout conditionally renders the navbar, homepage form, testimonial section, team section, and footer depending on the current controller and action, and whether the user is signed in or not

app/views/wishlists/index.html.erb

- Used current_user.email to display the current user's email in the page title.
- Created a container that displays the user's wishlist items.
- Used row-cols-1 row-cols-md-3 to define how many columns should be displayed on the page.
- Used a loop to iterate through each wishlist item and display the details in a card format.
- Displayed the item's title, image, and price.
- Added two buttons for each item: "Purchase" and "Remove from Wishlist".
- Used link_to to create clickable links for both buttons.
- Added target: :_blank to the "Purchase" button so that the link will open in a new tab.