CE219 Assignment

Creating a 3-tier web application for a car dealer

Report

James Lowry 1502558

1. Data analysis

1.1 Brand entity

Brand is the simplest entity in the project, having only four attributes, brand_id, name, nation, and cars. ‘Brand_id’ is the primary key for Brand and is incremented automatically with each new entry in the Brand table. ‘Name’ is the name of the brand, such as ‘Fiat’, ‘Ford’, etc. ‘Nation’ is the nationality of a brand, and ‘cars’ is the set of cars belonging to a brand.

1.2 Car entity

The car entity has six attributes: car_id, brand, model, price, available, and orders. ‘Car_id’ is the primary key which automatically increments with each new car; ‘brand’, of type Brand, is the brand to which the car belongs; ‘model’ is the model of the car, such as ‘500’ in ‘Fiat 500’; ‘price’ is the price of the car; ‘available’ is the number of cars that are available for sale; and ‘orders’ is a set of order entities for the car.

1.3 Customer entity

The customer entity has five attributes: cust_id, name, surname, phone, and orders.’Cust_id’, as per the other entities, is an auto-incrementing primary key for customer; ‘name’ is the first name of the customer; ’surname’ is the surname of the customer; ‘phone’ is a string representing the customer’s phone number; and ‘orders’ is a set of orders placed by the customer.

1.4 Order entity

The customer entity has five attributes: order_num, cust, cars, total, and pdate. ‘Order_num’ is the primary key which auto-increments with each new order; ‘cust’ is the Customer entity that placed the order; ‘cars’ is a set of cars ordered by the customer; ‘total’ is the total price of all the cars in the order; and ‘pdate’ is the date of purchase for an order.

2. Function Analysis

2.1 **index()**

Used to return the template for the homepage of the website.

2.2 all_brands()

Retrieves and returns all brands in table Brand, along with the template to view all of the brands.

2.3 show_brand(id)

Shows information specific to the brand represented by ‘id’ by retrieving that brand from the Brand table and returning it along with a template to view the information

2.4 edit_brand(id)
Retrieves and returns specified brand with a template for the form to edit said brand.
2.5 update_brands(id)
Retrieves brand, specified by id, to be edited. Edits each field with input obtained from form created by function 2.4.
2.6 add_brand()
Returns template for form to add a new brand to the Brand table.
2.7 new_brand()
Using the information obtained from the form created by function 2.6, creates and adds a new brand to the Brand table.
2.8 delete_brand(id)
Retrieves brand with brand-id = id, and deletes it from the Brand table.
2.9 find_brand()
Searches for a brand first by name, then by nationality,and returns any relevant results, or an empty list if no results are found, and a template for viewing the search results. In the event of an exception, an empty list is returned with the template for viewing.
2.10 all_cars()
Returns all cars in Car table and template to view all of them in a table.
2.11 show_car(id)
Retrieves car specified by id from Car table and returns it along with a template to view it.
2.12 edit_car(id)
Retrieve car specified by id from Car table, all brands in the Brand table, and return both with a template to display the car's current information along with possible brands that the car could belong to.
2.13 update_car(id)
Retrieve car by specified by id from Car table and set its attributes to the values obtained from the form displayed by function 2.12. Once attributes have been set, redirects to the show page for that car.
2.14 add_car()
Get all brands from the Brand table, and return them along with template to view form to add car to Car table.
2.15 new_car()
Creates new car with brand attribute corresponding to an existing brand, and saves it to the Car table.
2.16 delete_car(id)
Deletes the car indicated by id from the Car table.
2.17 car_search()
Obtains a keyword to compare against a car's brand, model, and price, and returns any cars that match the keyword in one area or another. If nothing is found, or an error is thrown, an empty list will be returned.
2.18 all_customers()
Returns all customers in Customer table along with template to view them all.
2.19 show_cust(id)
Returns customer whose cust_id matches id, along with a template to view the customer's information.
2.20 add_cust()
Returns the template for page to display form for adding a customer to the database.

2.21 new_cust()
Uses information obtained from form created by function 2.20 to create a new customer with a name, surname, phone number, and an initially empty set of orders, afterwards directing the user to the customer index page to view all customers.
2.22 edit_customer(id)
Retrieve and return customer indicated by id along with a template to view the customer information in form.
2.23 update_cust(id)
Use information obtained from form to update the customer's name, surname, and phone number. Afterwards redirect the user to the show_cust() page for the customer that was updated.
2.24 delete_cust(id)
Retrieves customer with cust_id = id from Customer table and deletes it from the table. Afterwards, redirect to the all_customers() page.
2.25 search_cust()
Attempts to find a customer in the Customer table using a customer's name and surname. Returns the any customer's matching the key obtained from a form, or returns an empty list if an error is thrown, or there are no results.
2.26 all_orders()
Retrieves and returns all orders from the Order table, along with a template to view all the orders in a table.
2.27 show_order(id)
Retrieves the order which has order_num = id and displays its information to the screen.
2.28 add_order()
Retrieves all customers and cars from respective tables and returns them along with a template to add a new order using the existing customers and cars.
2.29 new_order()
Retrieve customer and cars bought from form and use to create new order, adding prices of the cars together to obtain total, and using datetime.datetime.now() to get the current date and time for pdate.
2.30 add_cust_order()
Allows the user to add a new customer within the add_order() form. Displays the form from which attributes will be extracted.
2.31 new_cust_order()
Creates a new customer using information from form created by function 2.30 and redirects the user to the add order page.
2.32 edit_order(id)
Retrieve and return the order with order_num = id, all customers, all cars, and a list of the ids of the current cars in the order, along with a template to view the current information of the order.
2.33 update_order(id)
Retrieves order from Order table where order_num = id, a customer from the Customer table matching the one indicated in the form, updates pdate to current date so indicates the time order was last edited. Increments availability of all cars in the original order before clearing that table. Then uses form info to add new, possibly same, cars to the order's car set and recalculate the order total. Once finished returns user to the show_order() page for the edited order.

2.34 delete_order(id)
Retrieves order where order_num = id from Order table and calls the delete() method of the entity to delete from the table.

2.35 search_orders()
Attempts to find orders where the name/surname of the customer who placed the order matches the key obtained from the search form. The key can also indicate an order's order_num. Any order which has information matching the key is returned to be displayed as search results. If no results are found, or an error occurs, an empty list is returned.

3. Testing

3.1 Create brand test
No issues arose when creating the brand.

3.2 Add cars to system test
The application crashed when the tester tried putting a string in the price and availability sections of the form, but the tester was quick to realise the issue. After this test, errors of this kind are now caught.

3.3 Place an order test
The only issue that arose was that the customer accidentally submitted the form with incorrect information and hit the back button to see the form again after having saved the order. Once the information was fixed, the order was submitted again, leading to two order instead of only one.

3.4 Additional feedback
The website was **"**clear, straightforward, and easy to follow.**"** **–** Pauline Garrait