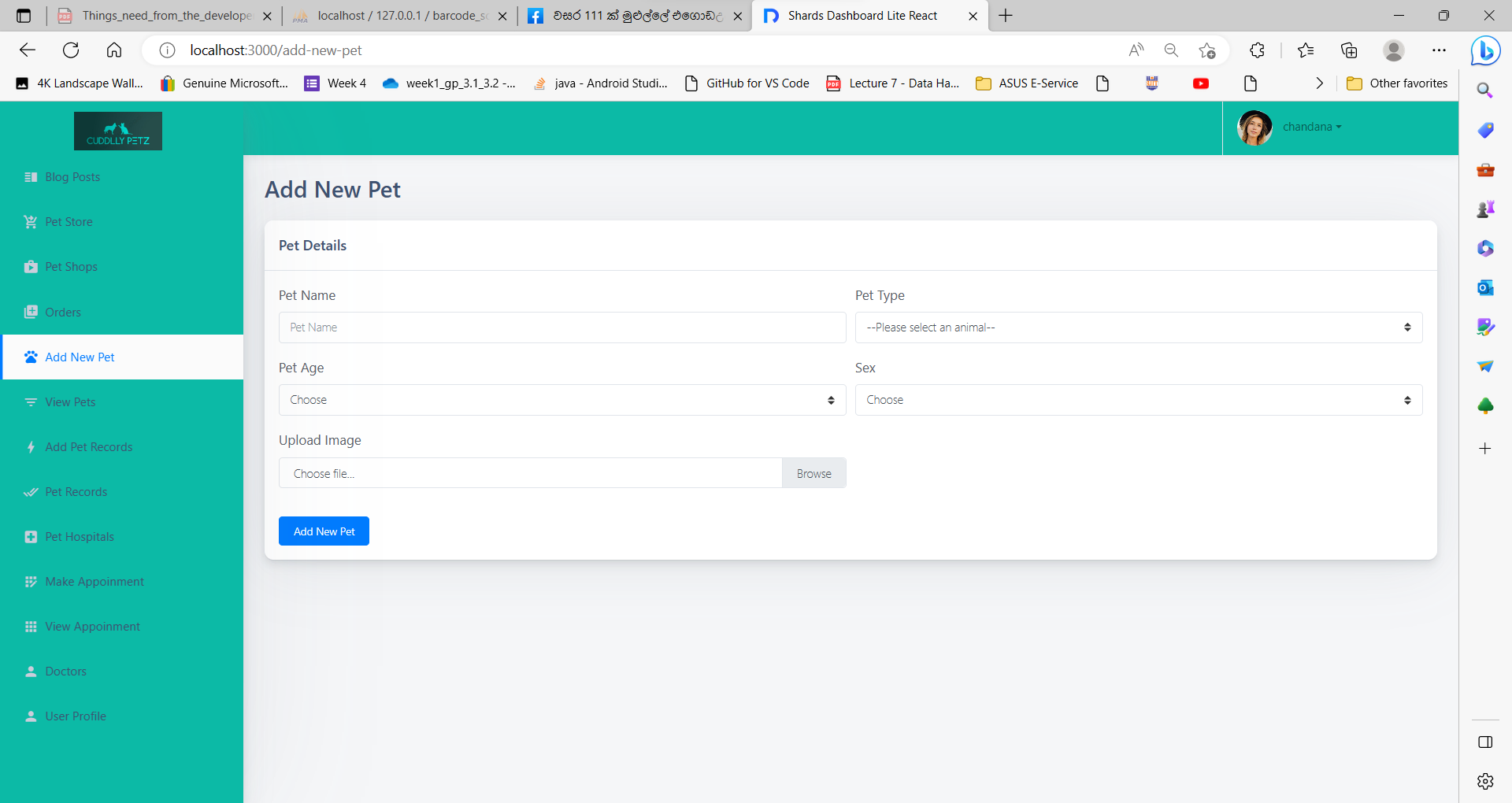
1)

Users should be able to create multiple accounts for different pets



Text

Description automatically generated

Text

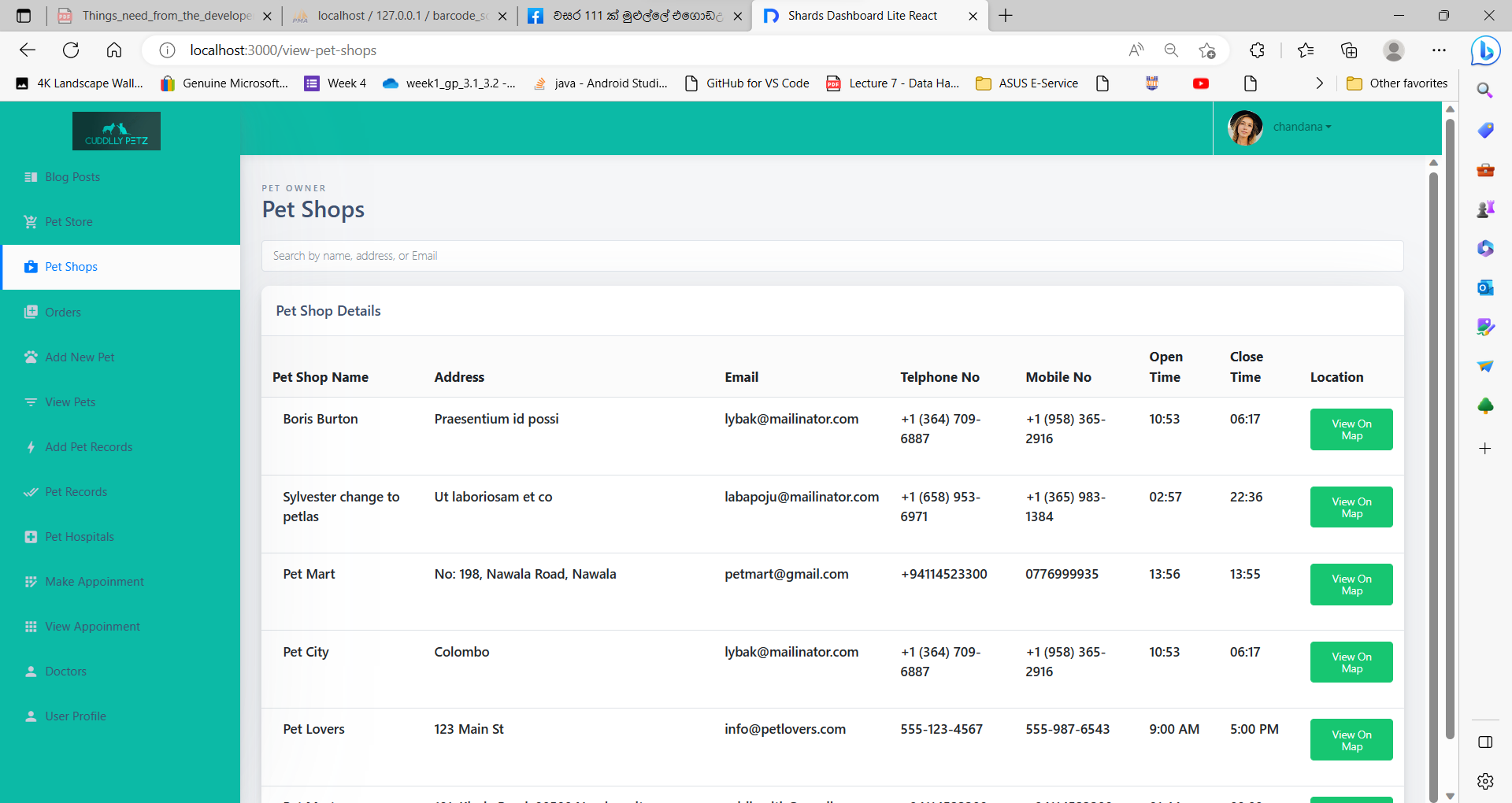
Description automatically generated

The process of obtaining new pet details through a form and storing them in a database via a REST API can be described as follows:

* User Interaction: The user interacts with a form where they can enter pet details such as PetName, PetOwnerID, SelectedType, SelectedBreed, Age, Sex, and avatar.
* Submitting the Form: Once the user fills in all the required pet details, they can press the "Add New Pet" button to submit the form.
* Creating a JSON Object: Upon form submission, the entered data is collected and used to create a JSON object. The JSON object will contain the pet details provided by the user.
* Sending Data to the REST API: The created JSON object is then sent to the REST API's body section. This API is responsible for handling the request and processing the data.
* Storing Data in the Database: The REST API receives the JSON object and extracts the pet details from it. The extracted data is then stored in the database. The database serves as a persistent storage for pet information.
* handleFormSubmit Method: The process of adding pet details to the database is implemented in the "handleFormSubmit" method. This method is called when the form is submitted, and it takes care of sending the pet details to the REST API for storage.

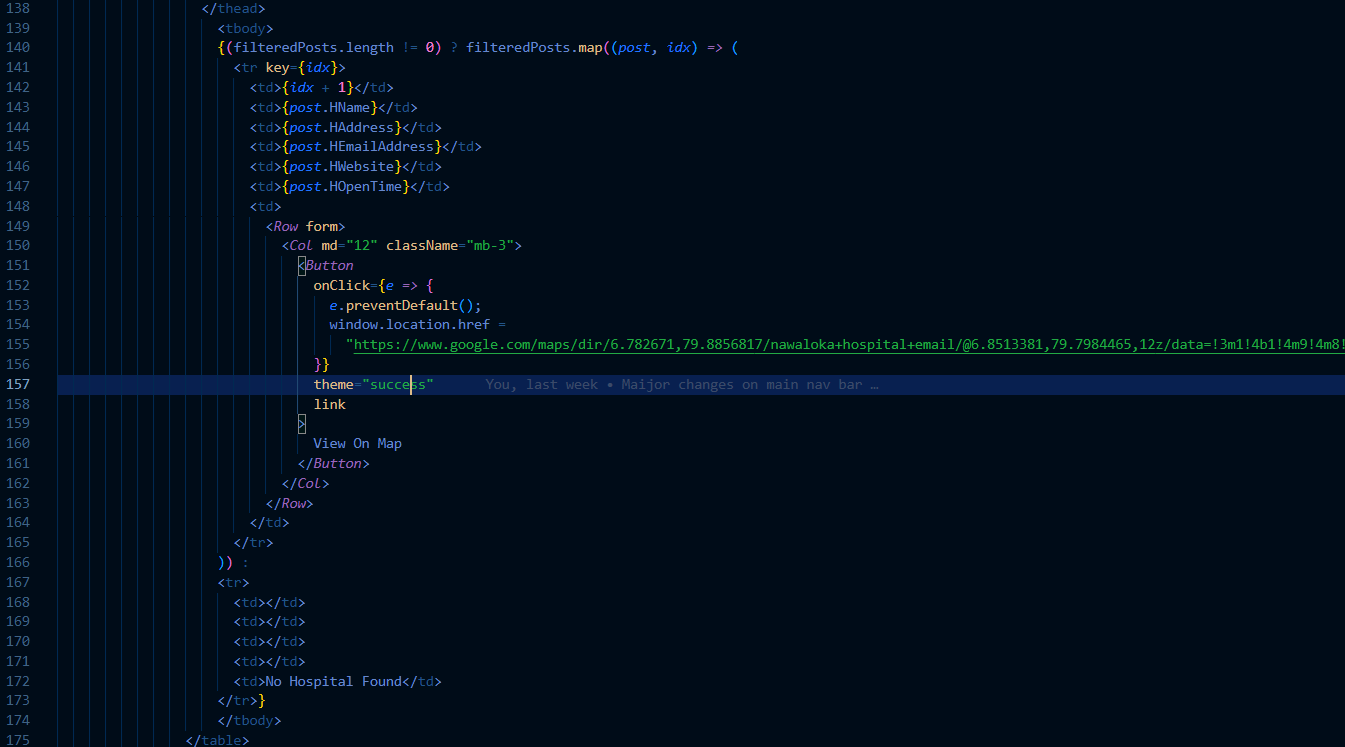
2)

Based on the shop/Animal hospitals or clinics the user is searching for, Users should be able to view the ones nearest to their location.

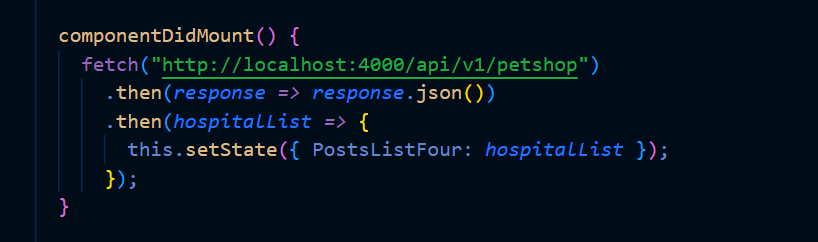


Graphical user interface, application, email

Description automatically generated





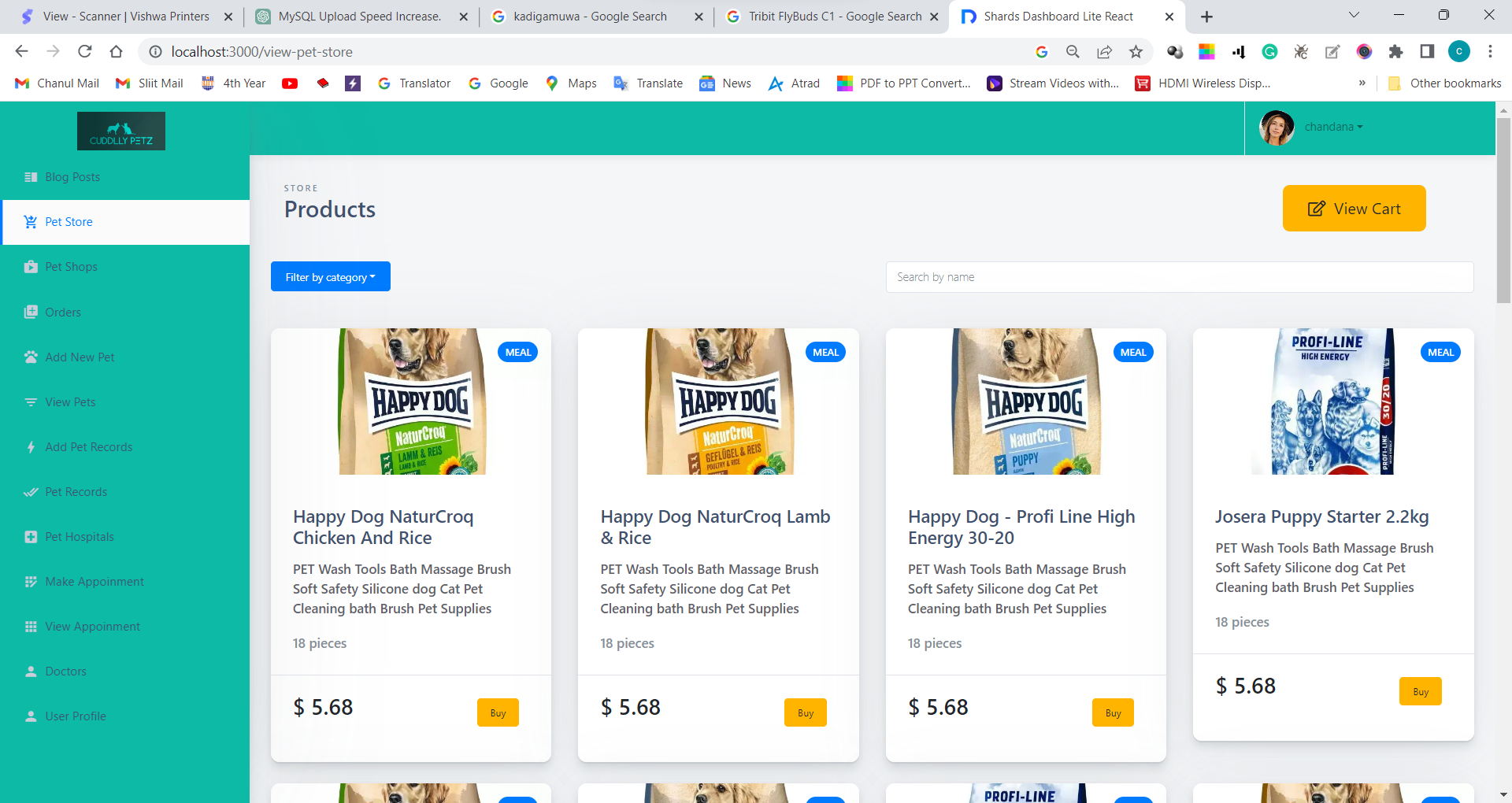


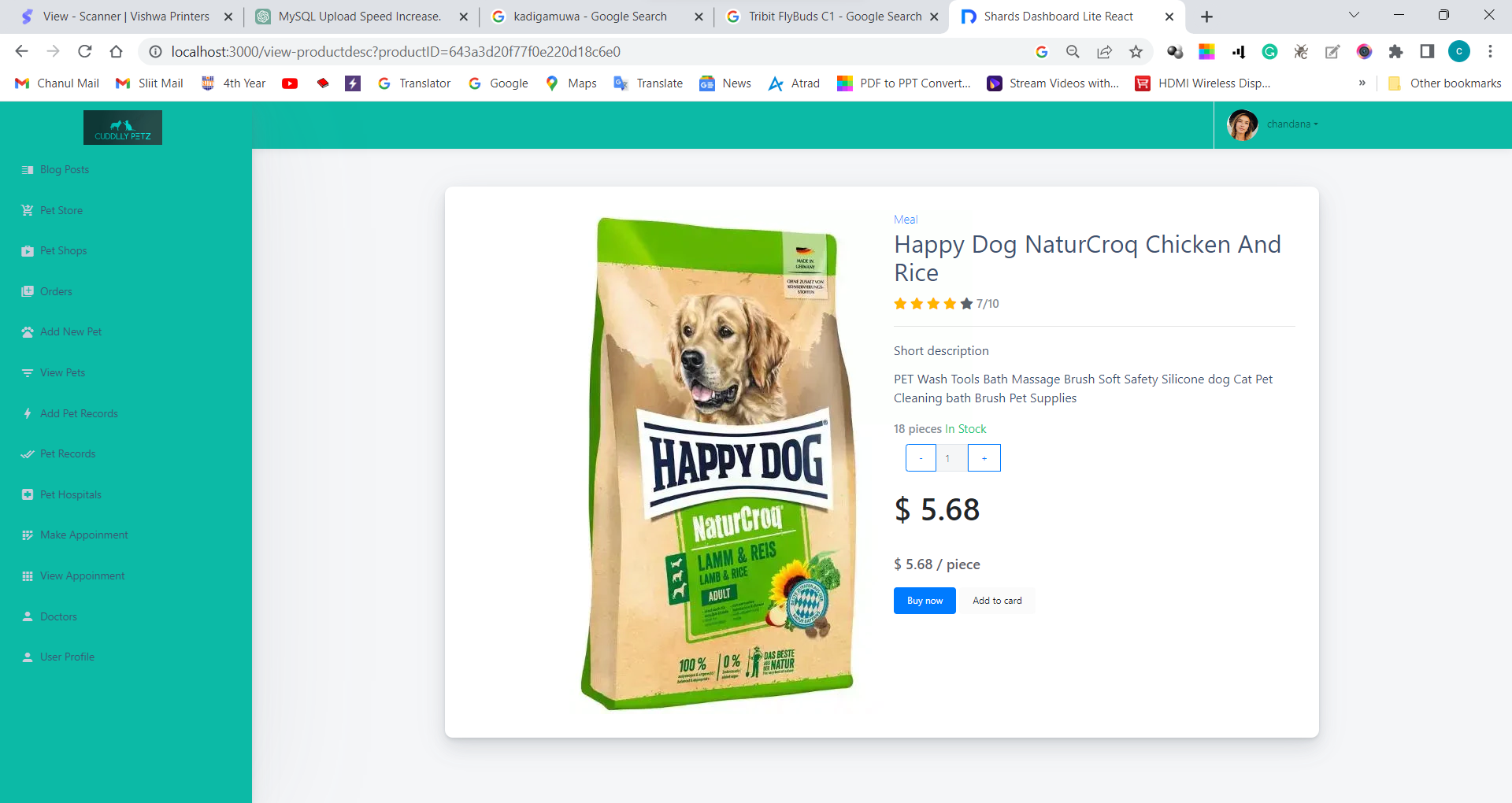
The functionality of allowing pet owners to view pet hospitals and pet shops can be summarized as follows:

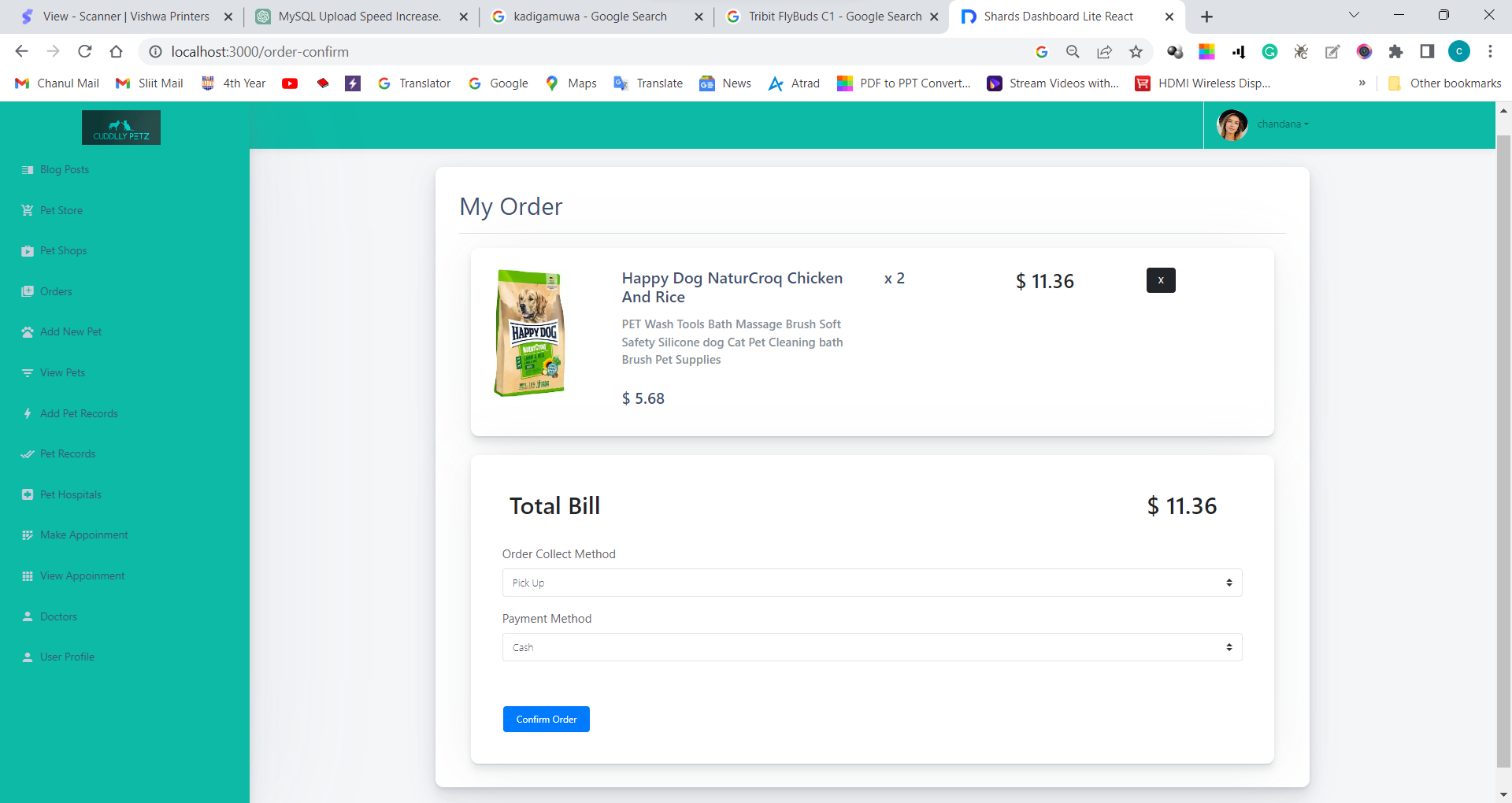
* Retrieving Data from the Database: The pet owner initiates a request to retrieve information about pet hospitals and pet shops from the database. This request is made through a REST API, which returns the data in JSON format.
* Storing Data: Upon receiving the JSON data from the REST API, it is saved in the variable named "PostListFour". This variable holds the information about pet hospitals and pet shops.
* Displaying Data in a Table Format: To present the data to the pet owner, the "map" function is utilized. This function allows iterating over the elements in "PostListFour" and displaying the details in a tabular format. Each row of the table corresponds to a pet hospital or pet shop, and the columns represent different attributes such as name, location, etc.
* Searching for Pet Hospitals/Pet Shops: The pet owner is provided with a search functionality to find specific pet hospitals or pet shops based on their name and location. They can enter the search criteria, and the system filters the data accordingly, displaying only the matching results.
* Viewing on Map: If the pet owner wishes to view a particular pet hospital or pet shop on a map, they can select the "view on map" option. This action triggers the system to display the location on a map interface.
* Directions to the Location: In addition to viewing on a map, the pet owner can obtain directions to the selected pet hospital or pet shop. By clicking on the respective option, the system provides navigation instructions to guide the user to the desired location.

3)

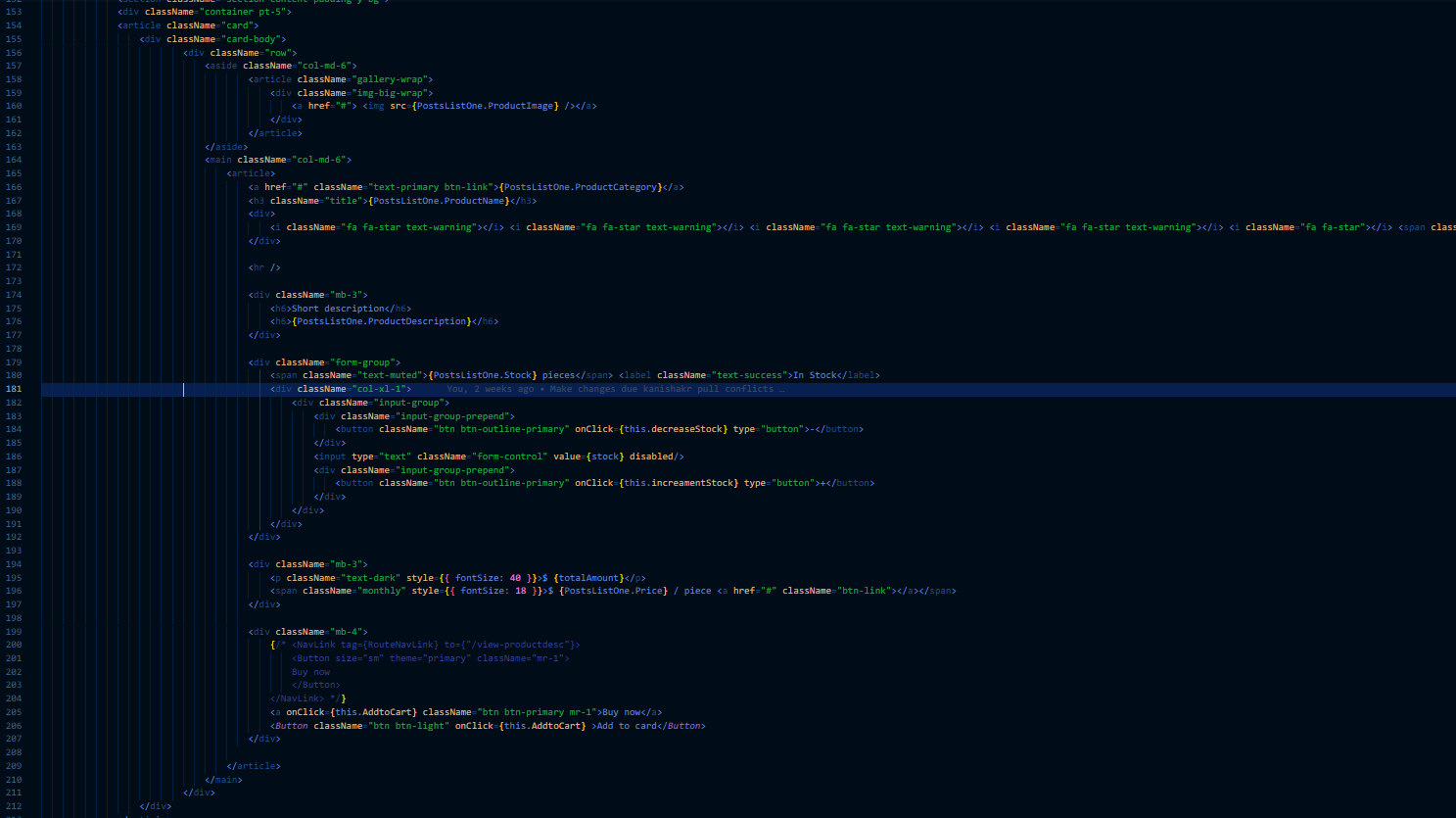
Users should be able to reserve products online.













“”

The product listing page functionality can be described as follows:

* Retrieving Data from the Database: The product listing page makes a REST API call to fetch product data from the database. The response from the API is stored in the "PostListOne" variable, which holds the product details in JSON format.
* Displaying Product Details: Using the "map" function, the product details are rendered in a card format on the page. Each card represents a product and displays relevant information such as name, price, and an image. This allows users to browse and view the available products.
* Filtering Products: Users have the option to filter the products by name. They can enter a search term, and the system filters the displayed products based on the matching names, providing a more focused view of the products.
* Product Description Page: When a user selects a product, they are directed to the product description page. Here, they can view all the details of the selected product, including a larger image, description, and other relevant information.
* Quantity Selection and Total Calculation: On the product description page, users can select the desired quantity of the product. As the quantity is adjusted, the system calculates the total amount based on the selected quantity. The calculated total amount is stored in the "TotalAmount" variable, reflecting the updated price based on the quantity selection.
* Order Placement: After selecting the quantity, users have the option to either "Buy Now" or "Add to Cart." Clicking either of these buttons directs them to the order confirmation page.
* Order Confirmation Page: The order confirmation page displays the selected products and the total bill amount. The product details for the cart are loaded from the database through a REST API call. Using the "map" function, the cart product details are listed on the page, allowing users to review their chosen items.
* Placing the Order: On the order confirmation page, users can choose their payment and pick-up methods. Upon clicking the "Order Confirm" button, an API call is made to insert the order details into the order database.
* Updating Order Product Table: After successfully inserting the order, the system adds the associated products to the Order Product table using the "map" function. Each product is inserted with a reference to the last inserted order's ID (Order.\_id).
* Order Confirmation and Success: Upon completing the order and related product insertion, a success message is displayed to the user as an alert, indicating that the order was successfully placed.