Project Summary

In this project, we enhanced Pete's Pet Shop by incorporating 6 new features, enriching the functionality of TC's Pet Shop. Here are the newly integrated features:

1. **Feature:** Pet Registration with Photos (i.e. feature 1 mentioned in L6 project instruction)
   Modification: Brings a new tab to the webpage which direct users to the registration page. Introduces new data structures, a struct to structure pet data and a mapping from pet id to pet struct, enhancing data organization and accessibility.
   Frontend: Our frontend starts a new webpage, and on that page, we added four text areas for users to enter the information of the pet and file input for user to choose the photo for the pet they want to register on the blockchain. We also used bootstraps and customized html elements to make the webpage look better and a button for going back to the home page of petshop.
   Backend: Our backend takes in the information passed from the front end, and builds a new struct called Pet, which contains petId, name, breed, age, location and photo of the newly registered pet and store it in a dictionary called pets, where the key is the petId (for quick referencing) and the value to be the new Pet instance.
   Benefit: Enables users to easily add pet information themselves, expanding access to a wider range of pets available for adoption.
2. **Feature**: Ether Donation (i.e. feature 3 mentioned in L6 project instruction)
   Modification: Brings a new tab to the webpage which directs users to the donation page.
   Frontend: Our frontend starts a new webpage, and on that page, we added an area to show what is the balance on the user's account right now and also a instruction of how to donate the money to PetShop. We also set the minimum donation to be 0.1 ETH and we set a checker to prevent user to enter a number that is less than 0.1 as well. On the other hand, we also created a table to show all the donations that are made to our PetShop including the information of who is the donor and the amount they donated. We also used bootstraps and customized html elements to make the webpage look better and a button for going back to the home page of petshop.
   Backend: Our backend serves three purposes here, first, it handles the functionality of storing the history of all donations in a customized struct called Donation which contains the information of donors' addresses and also the amount they donated in the one-time donation. Secondly, it handles the donation of the user by taking in the information of the donor and the amount of donations, store it to the struct and send the ether to the petShop's owner(we set the petShop owner to be the second address in our Ganache for easier access). Thirdly, it also performs the property of retrieving the donation history and providing it to frontend for rendering the result.
   Benefit: Facilitates ether donations to the pet shop via Web3Basics SendMeEther integration, ensuring secure and transparent transactions. helps the pet shop obtain additional funding avenues to support its operations and facilitate more pets in finding new homes.
3. **Feature:** Pet Filtering (i.e. feature 5 & 6 mentioned in L6 project instruction)
   Modification: Frontend: For the home page we added three select bars for user to select from three categories: Bree, Age Range and Location of the pet. All the information in Breed, Age Range and Location

selectors are fetched from the local file and the information on the blockchain. So, after adding a new pet, the selector's options will change. Also, we implemented these three selectors in an "And" relation which means it only shows the pets that qualify all the filters that user chose. And the Result will render immediately after using make the selection.

Backend: The backend is not doing anything here, because the pet's information was already retrieved and fetched and saved on the frontend.

Benefit: Empowers users to filter and discover pets based on specific criteria such as breed, age, and location, enhancing the adoption process by matching pets with suitable owners efficiently. Streamlines pet selection, saving time and effort for both users and administrators.

4. **Feature:** Most Adopted Breed Tracking (i.e. feature 7 mentioned in L6 project instruction)
   Modification: Brings a new tab to the webpage which displays the pets of the most adopted breed.
   Frontend: For our frontend we added a link element to trigger the function to retrieve the most adopted pet Breed from the blockchain. After getting the result, using this result to filter and get the correct pet's from our fetched pets. Using this information to render the page. The webpage will render the result immediately.
   Backend: For our backend, we added a function called getMostAdoptedBreed() which will go through all the pet's information that store on the blockchain to count how many of pets are adopted for each breed. Then send the information back to the frontend for it to render the result.
   Benefit: Fosters informed decision-making for both prospective pet owners and the pet shop, potentially influencing future adoption strategies for both.
5. **Feature:** Adoption History Management (i.e. feature 10 mentioned in L6 project instruction)
   Modification: Brings a new tab to the webpage which displays all the pets once adopted by the user.
   Frontend: For our frontend we added a link element to trigger the function to retrieve the user adoption history from the blockchain. The webpage will render the result immediately.
   Backend: We created a mapping from user address to an array of pets. When a user adopts a pet, the pets data will be pushed to the mapping. Once getAdoptionHistory() is called, we return data from the mapping.
   Benefit: Allows users to maintain a comprehensive record of their adoption history, including pets they have previously adopted and subsequently returned. Provides easier access to the status of users' past adopted pets and supports users in making personalized decisions.
6. **Feature:** Pet Return Mechanism (i.e. feature 13 mentioned in L6 project instruction)
   Modification: Frontend: Firstly, we added a button for each pet Template for triggering the function handleUnadopt() it gets the petId of the pet that user wants to return and sends it back to backend for it to perform the unadopted functionality. Then it will re-render the webpage to refresh the button. We also make some changes to markedAdopted function make it be ablet to mark the pet correctly when the pet has no adopter the adopt button will be enabled, if the pet was owned by the user the unadopt button will be enabled and if the pet is own by others both buttons will be disabled. This information was provided by App.pets which is updated and fetched from blockchain and the local file.
   Backend: Our backend handles the unadopt for on the blockchain, it takes the petId that the user wants to unadopt from the frontend. Then it finds the pet in pets dictionary and marks its owner to be Empty.
   Benefit: Provides flexibility and reassurance to pet owners, ensuring the safety of pets regardless of the reasons behind the owner's decision to relinquish them.