Version 3.0

Task 0: Explain what you are doing/ going to accomplish

I am going to design the personal pages for each dog as well as add in the “rent” functionality.

Task 1: Sketch interface design



Her we have a large main picture with all the information neatly laid out next to it, there should also be a rent button somewhere, however I have not yet decided where I will place it. Either in the center box, or below it. Or at the top in the center? That will come with the development of the program.

Task 2: Identify any classes required

Class Dog

Will need the variables, Name, Age, Description, Breed, Available, Gender

Task 3: Identify information to be displayed

This will need to display the dogs information as well as whether they are in use or not.

Task 4: Identify user inputs

Clicking on the “rent” button will set the dog as “used” for a day

Task 5: Identify any constants or existing data if required

No constants for this version

Task 6: Identify indexed data structures

Data dictionary called dog\_list

Task 7: Determine what calculations are necessary

No calculations needed in this version

Task 8: Develop a modular structure for your program

 Set route to /dog-page/<id>

Set view to dog-page

Def dog-page:

Set dog\_id equal to int(dog\_id)

Set found dog to none

For dogs in dog\_list:

If dog.id = dog\_id:

Set found dog to dog\_id

Break the loop

Set data to a dictionary of the dogs info

Return Data to the page

Task 9: Define the functions identified

Dog-page defines each personal dogs pages

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

No copyrighted images. No illegal or explicit images etc.

Again in this version I use purple colours to make sure you can tell the important parts. The stars, as well as the buttons are shown in purple to bring attention to them.

Task 11: Document test cases for testing the program

Load, “localhost:8080/dog-page”, and see if the page show up to how I imagined it within the interface design sketch.

Also try renting a dog and see that it becomes “unavailable” after it has been rented out.

Task 12: Refine the plan

I had forgotten to assign each dog an ID so the page was not working, I Fixed this by adding in the line of code

self.id = next(self.\_ids)

This assigned each dog a unique ID and allowed the page to work.

Within my html I added the code **% for star in range(dog.friendliness)** so that I could show a variable amount of stars for the dogs friendliness rating system.

And then **% for star in range(dog.friendliness, 5):** so that I could fill up the abr to make it out of 5 stars.

This allowed a simple yet effective way of rating each dogs friendliness.

I need to make a doggy rent success page so that the user knows they have rented the dog.

I will need to add new code to the python file to do this.

Pseudocode.

Set route to /dog-rent-success

Set view to dog-rent-success

Define rent\_success to:

Set dog\_id equal to int(dog\_id)

Set found dog to none

For dogs in dog\_list:

If dog.id = dog\_id:

Set found dog to dog\_id

Break the loop

Set data to a dictionary of the dogs info

Set found\_dog.availability to 0

Return Data to the page

To add to the program, I wanted to add in a date so that the time the dogs were taken out could be shown to other users. As well as a contact form to send to the “owner” of the dog.

In code this required me to add a variable to the dog class called, “date”

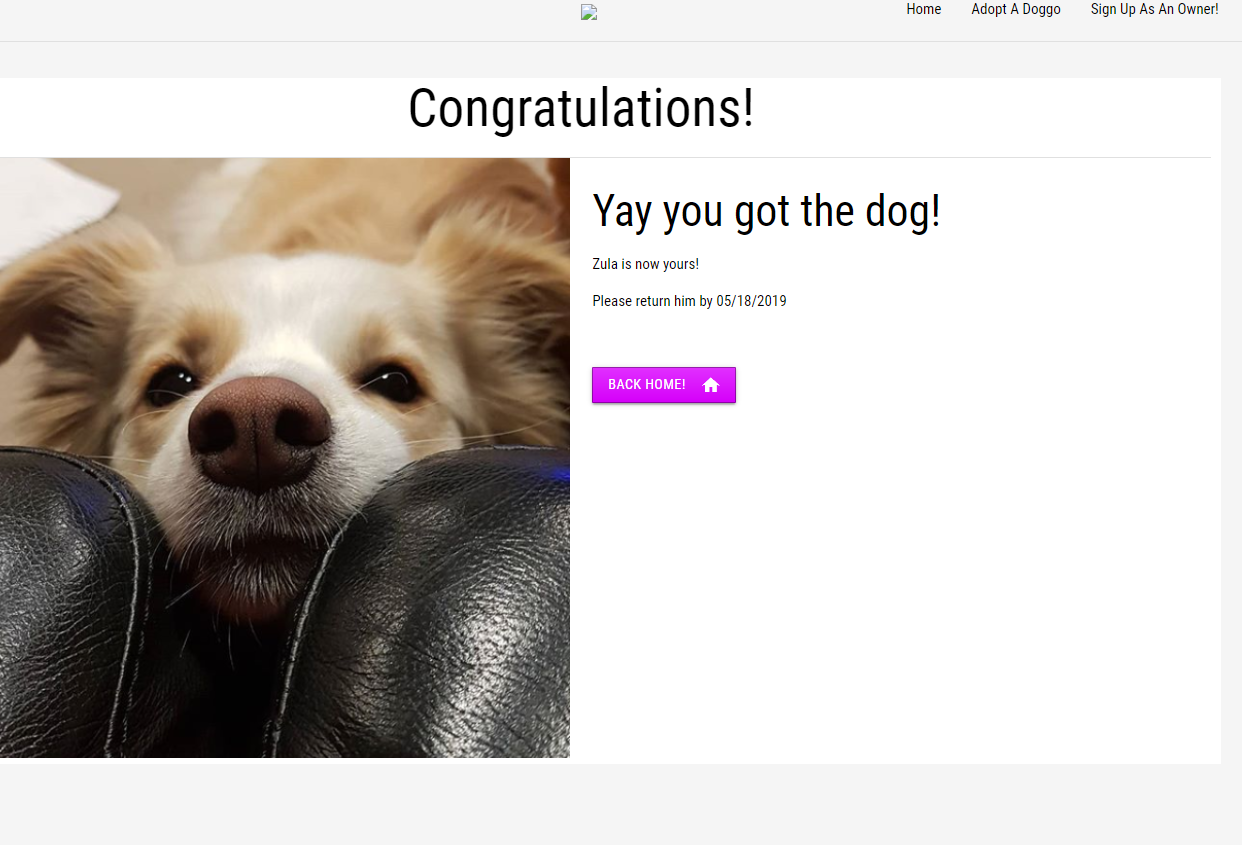
Import datetime

In the rent\_success code I add in,

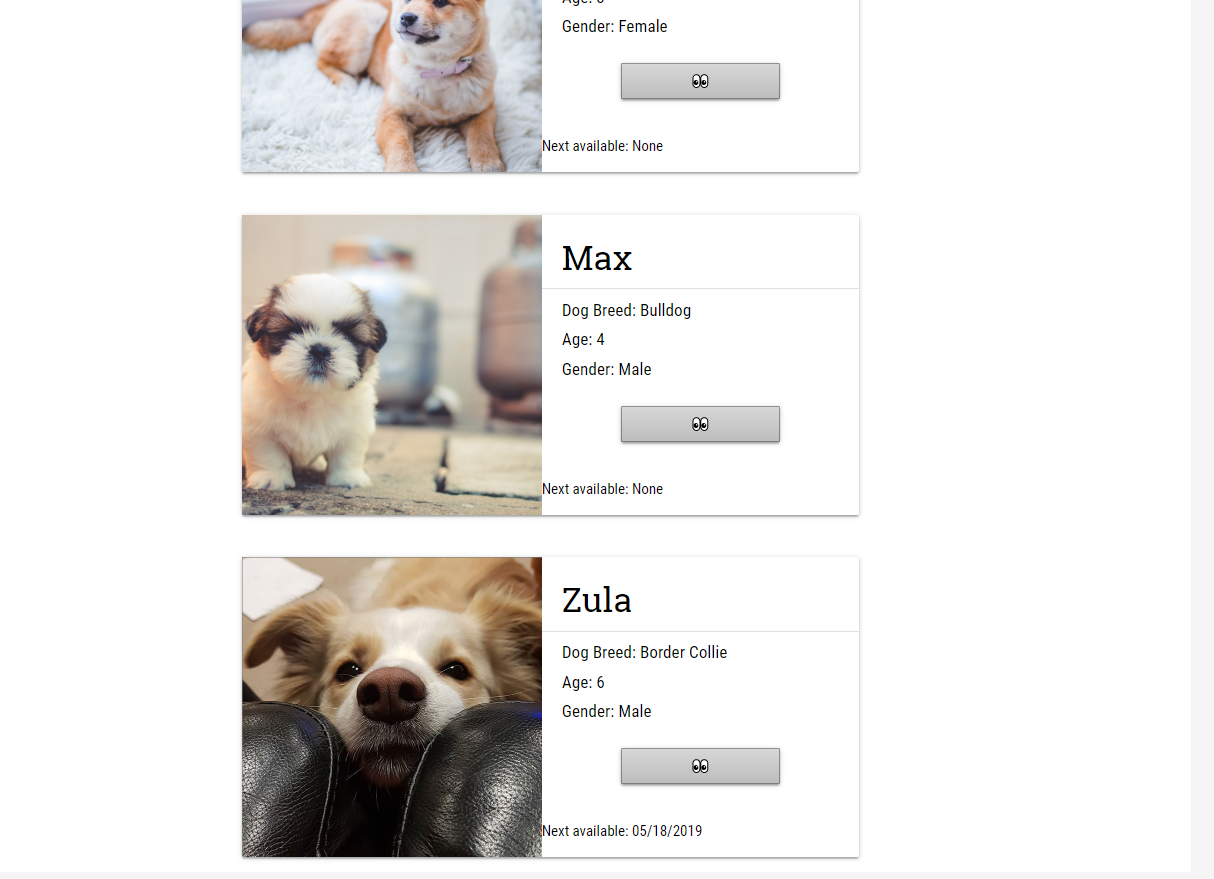
Set found\_dog.date to be datetime.now()

In the showcase page for unavailable dogs, add in below the button the date the dogs are a vailable {{dog.date}} + timedelta(days=1)

Task 13: Document testing



 This shows the successfully rented page!



And how after being taken, the dog become unavailable

Task 14: Evaluation

This version worked well. Still need to complete the aesthetics of the page, however the functionality for now is there. I’m working on getting achieved then adding excellence parts later