Version 6.0

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

I will add the ability to track the name of the person who “rented” out a dog as well as keep their email handy within their object.

Task 1: Sketch interface design

Not applicable for this version, no new pages created.

Task 2: Identify any classes required

Person class

Dog Class

Task 3: Identify information to be displayed

Dogs name

The users name

Task 4: Identify user inputs

Name, email address

Task 5: Identify any constants or existing data if required

No constants for this version

Task 6: Identify indexed data structures

List called Persons made up of people objects

Task 7: Determine what calculations are necessary

date = datetime.now() + timedelta(days=1)

This calculation adds 1 day to the date

Task 8: Develop a modular structure for your program

…

Set name to request.forms.get(“person\_name”)

Repeat for all variables

Set new\_person to Person class with above defined varibalbes

Set new\_person.dog to dog\_id

For dogs in dog\_list

If the dog.id matches dog\_id

Set found\_dog to dog

Break for loop

Set new\_person.dog\_name to found\_dog.name

…

Return data, human\_data

Task 9: Define the functions identified

None

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

No copyrighted images. No illegal or explicit images etc.

Within this version I need to make sure the form to enter your name is easy to fill out and clean. I can’t ask for too much information which is why I only ask for name and email. That is enough to one, name the user and two, to contact the user.

Again within this version I will ensure the same design features are used within the page, such as purple buttons and simple colour/design. To make usability as easy as I can.

Task 11: Document test cases for testing the program

When testing the program, does the date etc show up in the dog success page

Because the inputs within materialize already have a validation feature. There is no need to test the inputs. Emails must be a valid email to go through (they must include an “@” sign) and names must be a valid name. In this case I do allow numbers as you never know who might rent a doggo. <https://www.quora.com/Can-human-names-contain-numbers>

This site has examples of humans with numbers in their name, such as Bob 17, and 7.

Task 12: Refine the plan

I found that I was not able to return two dictionaries as it returned an error. Because of this I needed to change the way my program works and add the name of the owner into the name of the dog.

I added in the code within the human class.

Set self.dog to None

Set self.dog\_name to None

Set self.return date to None

I later found that return\_date = None returns an error so I changed the code to

Set self.return\_date to “0/0/0”

Which fixed the error as None does not work for date types. Which I only just found out.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

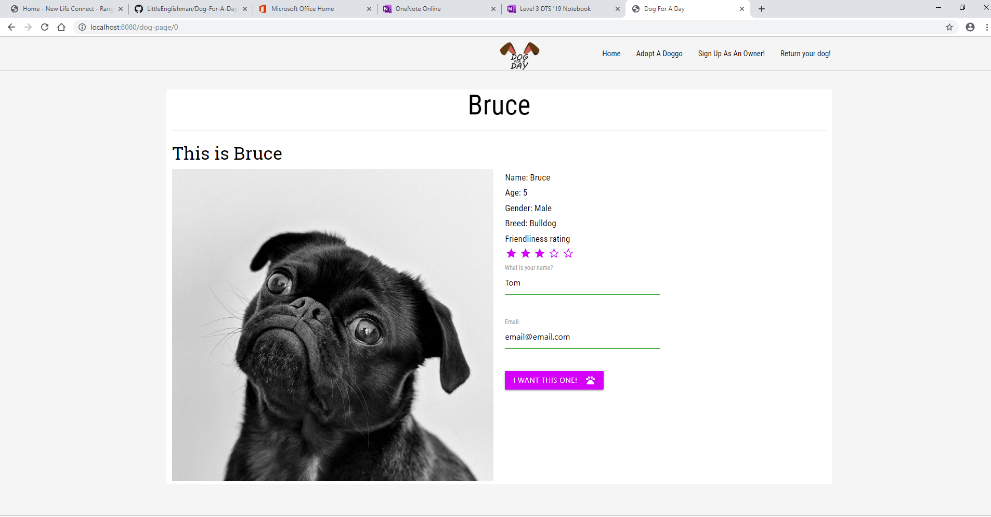
Because I could not return two dictionaries, I changed the return statement

Return human\_data

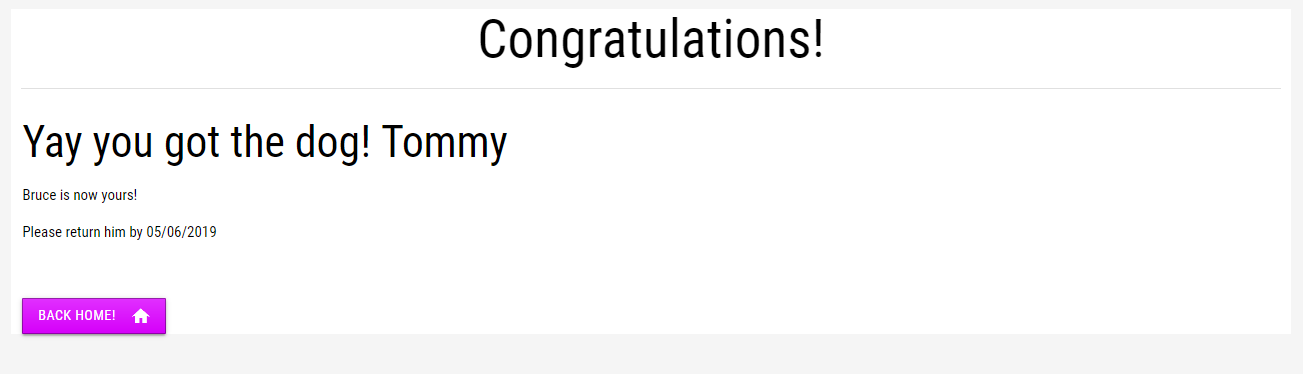
As I did not need the dog data due to the changes above.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Task 13: Document testing



Here you can see the new form that shows up when you load a personal dog page, asking you your name and email.



Here you see the name and date displayed on the congratulations page. Formatting is not complete yet though.

Task 14: Evaluation

This version took a lot of bug fixing/figuring out how to do it. But in the end I got it to work. It now works how I wanted it to. This version allows me to create a more “personalised” user experience which makes the program more robust and friendlier to the user. It only collects name and email as a way to ensure the users privacy is upheld.