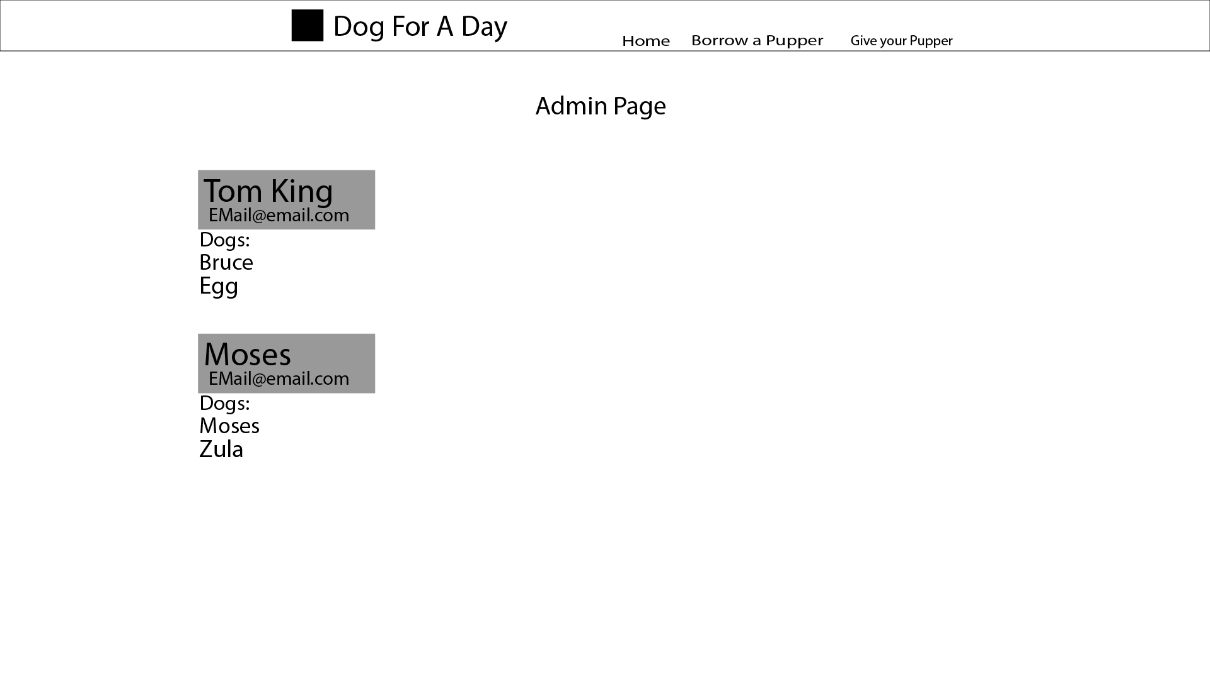
Version 8.0

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

I will create an “admin” page where the users and what dog they have rented out can be shown.

Task 1: Sketch interface design



This will be a simple page made for the admin. Thus, it requires no bells and whistles. It will follow the same design as the other pages

Task 2: Identify any classes required

Person class

Dog Class

Task 3: Identify information to be displayed

Persons name/email + dog 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 with the variables name, email and dog

Task 7: Determine what calculations are necessary

None

Task 8: Develop a modular structure for your program

Set route to “/admin-page”

Set view to “admin-page”

Define admin\_page as:

Set a dictionary of humans form the person list

Return that dicitonary

Task 9: Define the functions identified

Admin page, which define the webpage

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

Within this version, I will need to create a website interface that is easy to read and simple to understand as many of the users may be older/unfamiliar with technology. I also need to follow the general rules of design when it comes to websites with colours layout etc. Buttons need to be clear and laid out, everything should make sense.

It need to be functional, it should first fulfil its purpose and secondly look aesthetically pleasing,

No copyrighted images. No illegal or explicit images etc.

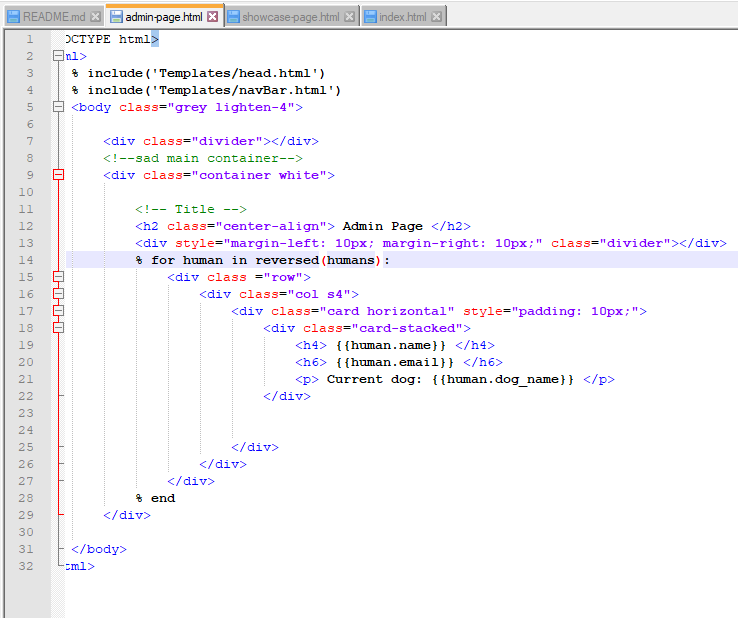
Task 11: Document test cases for testing the program

To test this program, first I will load localhost:8080/admin-page, to check the page shows up correctly. And then I will try taking out a new dog. And then seeing if the new renter turns up on the list or not. Than I will return that dog and check to see the dog doesn’t still appear on the list.

Task 12: Refine the plan

Everything worked first time surprisingly.

However, after consideration, I realised that over time, this list of past users would build up and up and you will have to scroll very far down to find the users that have a dog out currently. To do that I needed to reverse the list, so that the newest additions would be shown on the list first on the webpage.

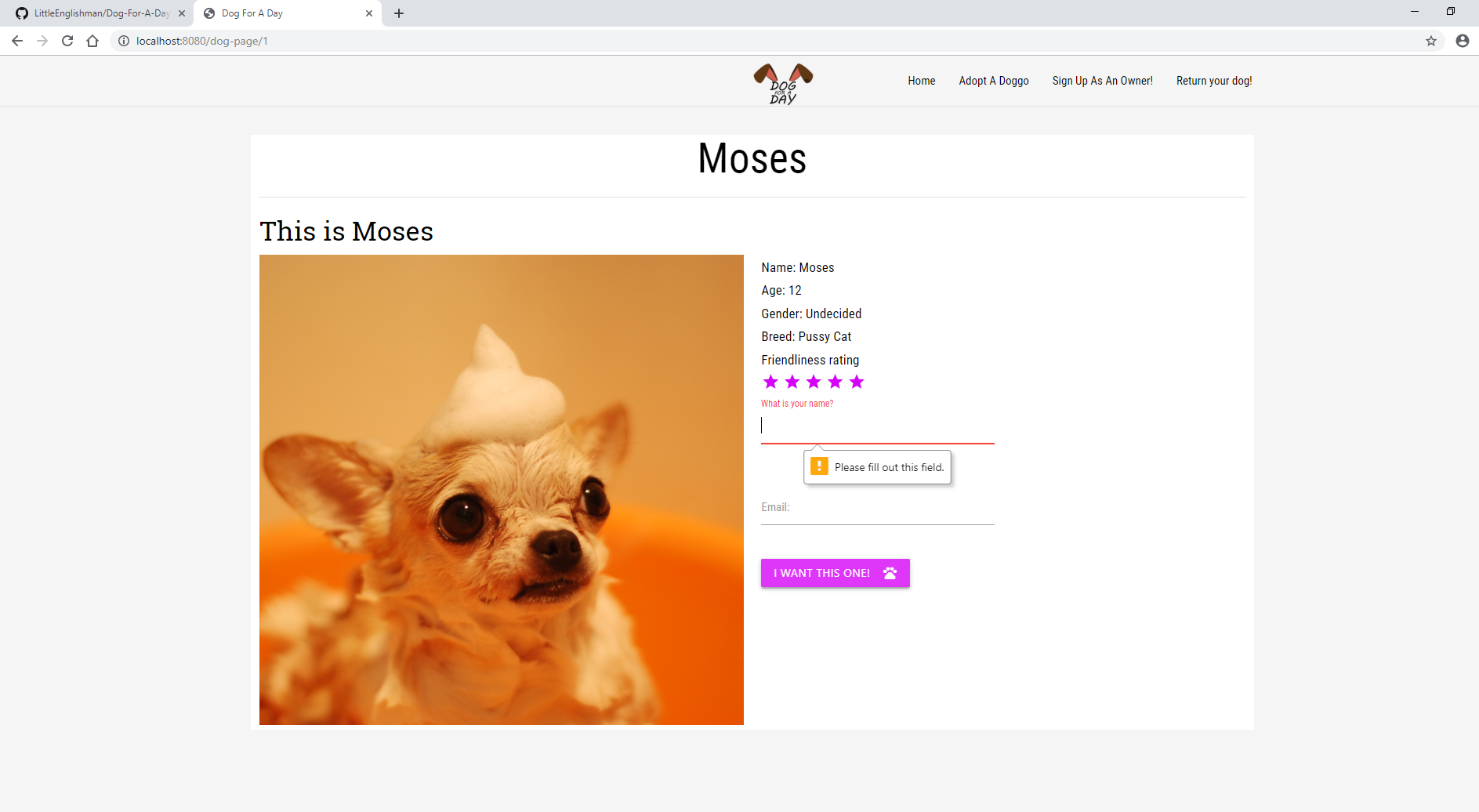


I did this by adding in the “reverse() code” that is selected above.

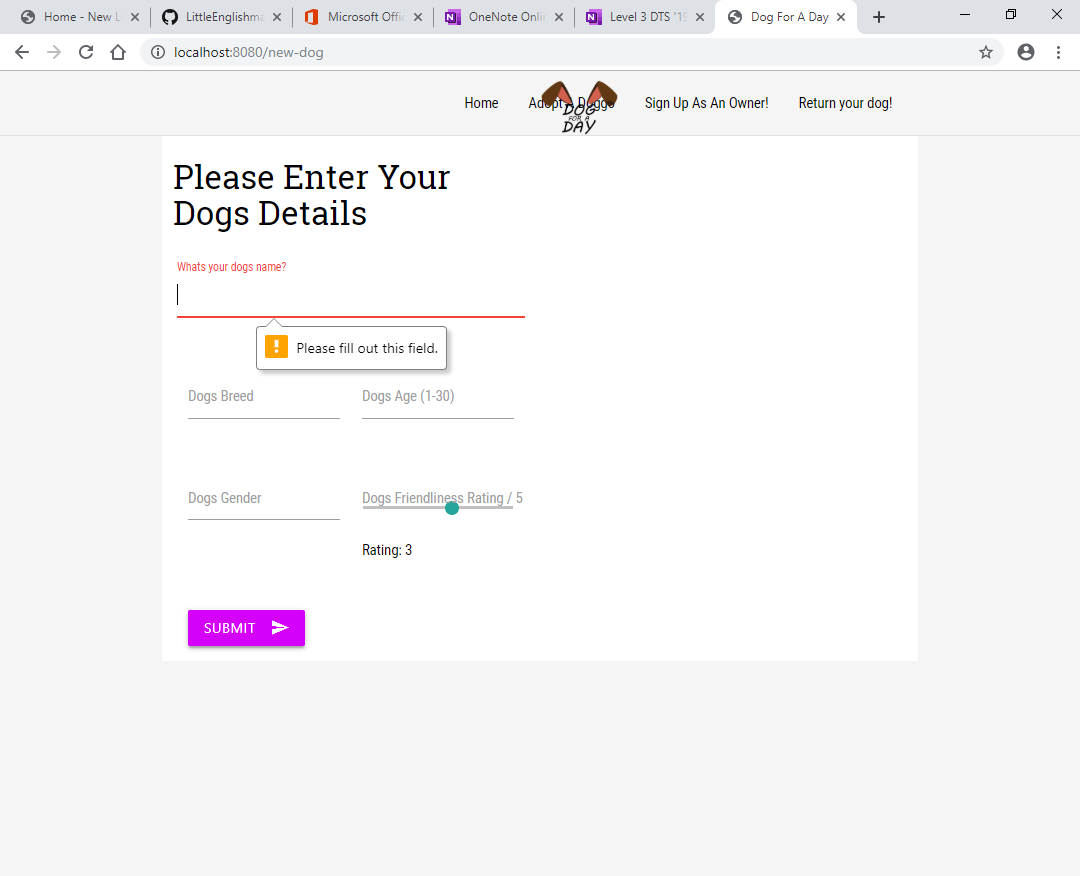
(<https://thispointer.com/python-different-ways-to-iterate-over-a-list-in-reverse-order/>) I found it on this website

That fixed the problem as it know shows the list in “reverse order”.

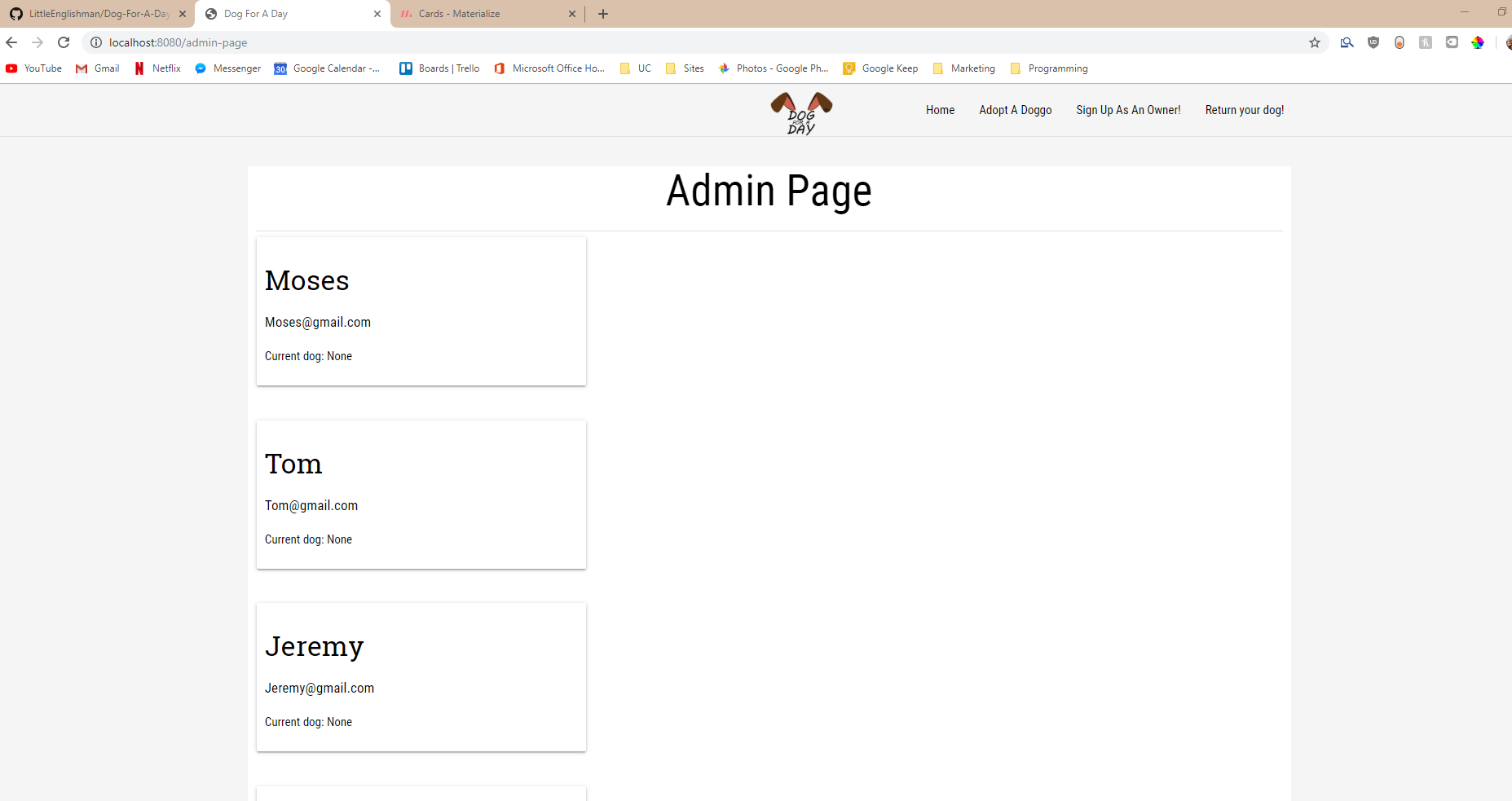
Throughout my testing, I also found out that I could input an empty form, leading to an empty buyer taking the dog. To combat this I added in the word “required” to each input in the form to make them necessary for the form to be submitable. This fixed the bug as forms are now not able to be submitted if empty



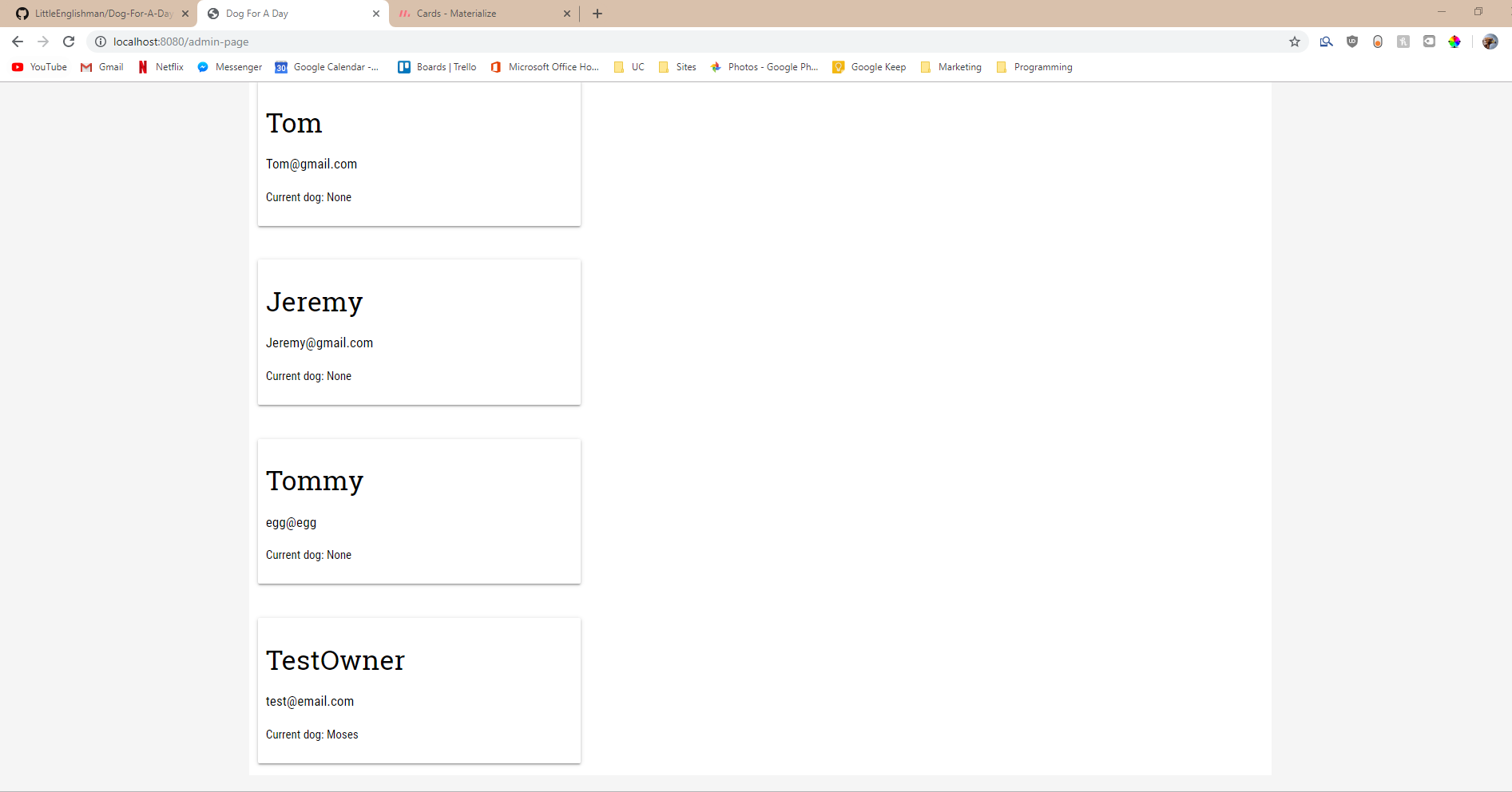
Here you can see the forms do not submit if empty as they show a relevant error message. Luckily most of this is built into materialize forms so I do not have to format it or code it in separately.



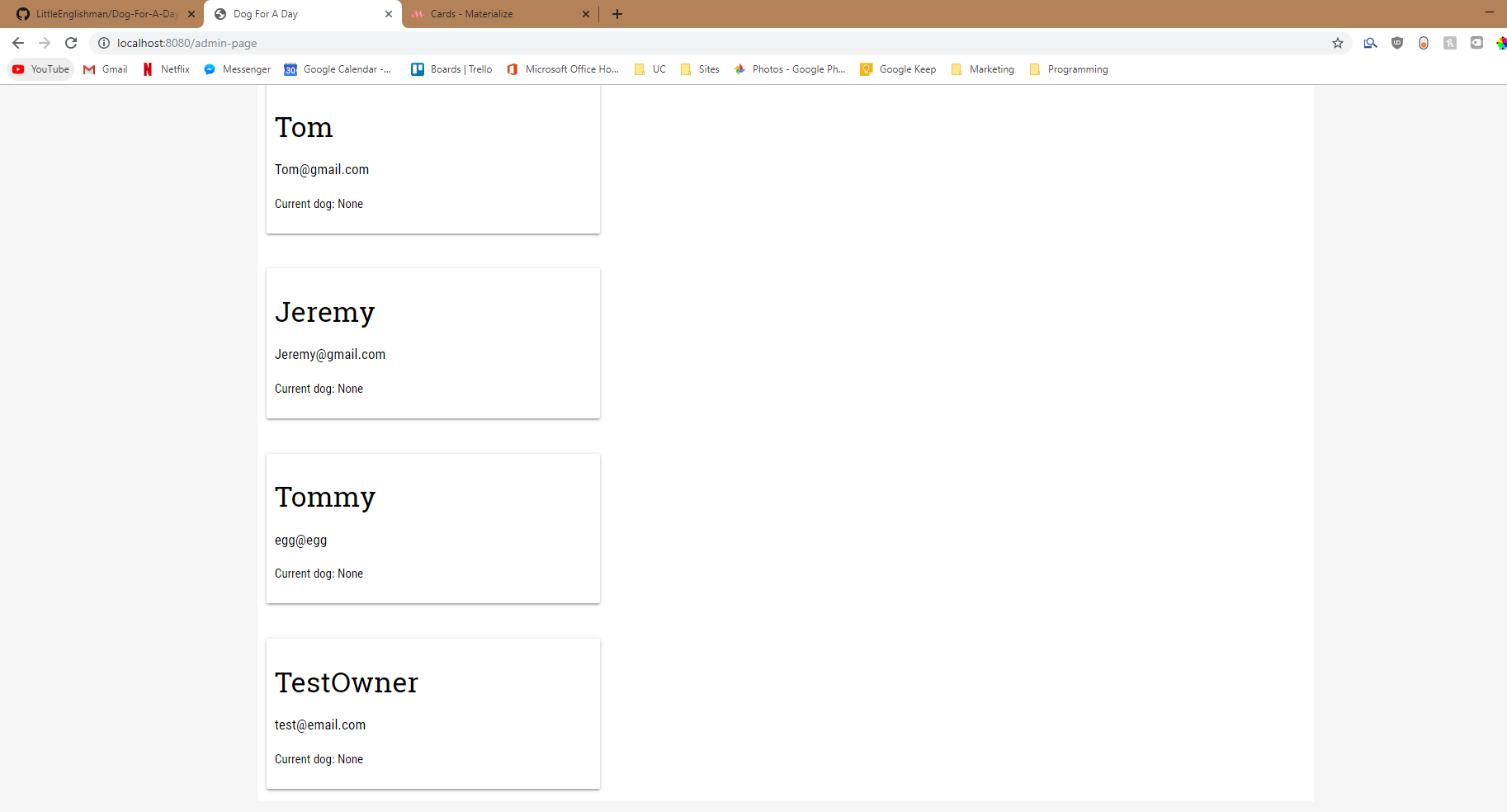
Task 13: Document testing



Here you can see the page loads with the test cases placed in. Dog shows up as none as they currently are just empty values.



Adding in a test owner works. Also shows the dog they have



When the dog is returned, the Current dog goes to none.

Task 14: Evaluation

This version was very successful and everything shows up as needed. This version allows an admin to see who has hired out which dog and contact them. It fulfils the need of an Admin page and works well but does not need to looks amazing. The change to make all forms required also fixes the problem of having empty forms which is a great fix and ups the functionality of the page.