Andrea Sanchez
Santiago Barrios
Alexandra Tabares
Bruno Montes

Sprint 3 Documentation

Detailed Work in Sprint 3:

Front-End:

Fixed a critical error in which our front-end in angular would not be able to compile due to invalid user access properties as one of our systems when developing the front-end had privacy settings in which if a user were to make changes to any already existing files, the error would occur due to invalid user access and thus even though our code and routing was correct, visual studio would not allow any changes made by any other user to be saved.

Created and tested our component cypress tests to ensure that our two test cases that failed in the previous sprint passed as the reason those tests failed was due to the pages not being able to be compiled and thus failed. However, now that the front-end is able to be compiled, the tests were able to be passed since they are now able to access the contents of the page.

Updated the “About” page to be more appealing and user friendly as before it was only place holder text displaying “About Works!”

Updated “Contact” page with user interactive text input boxes so that users are able to send us any information on bugs regarding the website. Plans to make this more user friendly and appealing just like the update done to the “About” page.

Back-End:

With the cat object, we are planning on making a quiz where the user can be paired with a cat based on the features of the cat. Therefore, each cat needed to have a certain feature to be able to be paired. We created a set of features and assigned them to each cat to give our user more personality and add another level of intractability to the website.

Reorganized the structure of the Backend by allowing one main entry point instead of having different entry points. We did this by reorganizing the files scrape.go, scrape_test.go, api.go, api_test.go into two folders: scrape and api. The result of this process allows us to use the data from scrape and import it through api.go and main.go; making the process a lot more streamlined to use.

Came up with a solution for an issue in the backend to display the information from scrape.go to Postman. This issue was resolved by creating the cat slices in each function inside api.go.

While this method works, we are currently trying to find a method to make one call instead of calling every time you're in a function.

Front-End Unit Tests:

OLD:
Cypress Component Test - Home

This unit tests for a valid routing when the user clicks on the “Home” button on the navigation bar on the webpage. - Failed (Now Passing in Video Demo)

Cypress Component Test - Shelters

This unit tests for a valid routing when the user clicks on the “Shelters” button on the navigation bar on the webpage. - Failed (Now Passing in Video Demo)

Cypress Component Test - Contact

This unit tests for a valid routing when the user clicks on the “Contact” button on the navigation bar on the webpage. - Passed

Cypress Component Test - About

This unit tests for a valid routing when the user clicks on the “About” button on the navigation bar on the webpage. - Passed

Cypress End-to-End Testing

This unit tests for a valid routing when the user clicks on the MAIN Page of website as well as other default end to end testing provided by Cypress. - Passed

NEW:

Cypress Component Test - Forum Access and Interactivity for Contact Page

This unit tests for the ability to click on and access each forum field as a way to verify that the input text box is interactable. We want to be able to have the the functionality of filling out the forum automatically. - Passed (Partially because we want the filling feature by next sprint)

Cypress Component Test - Button Functionality for Home Component

This unit tests for click functionality for the “LOCAL cats” button on the home page for the website. - Passed

Cypress Component Test - Button Functionality for Cat Component

This unit tests for click functionality for the “Adopt” button on the home page for the website. - Passed

Back-End Unit Tests and functions:

OLD:
*func TestInWebsiteMiamiDade(t *testing.T)*
*func TestInWebsiteLakeCounty(t *testing.T)*
*func TestInWebsitePeggy(t *testing.T)*
*func TestInWebsiteKeyWest(t *testing.T)*
*func TestInWebsiteMarathon(t *testing.T)*

The above five unit test functions test if the correct cats are being updated into the cat list. We created separate functions that convert the cat list into a string to be able to compare them in the unit tests. However, this can change because the shelters update their cats every day.

*func TestInWebMarathonAge(t *testing.T)*

This unit test function checks if the Age property of the cat object in Marathon Animal Shelter returns “In Website” since the HTML tags do not return the age of the cat for this shelter.

*func TestInWebLakeCountyBreed(t *testing.T)*

This unit test function checks if the Breed property of the cat object for Lake County Animal Shelter returns “In Website”. This test is needed because the Lake County Animal Shelter does not display the breed of the animal on the website.

NEW:

*func TestFeaturesMiamiDade(t *testing.T)*
*func TestFeaturesLakeCounty(t *testing.T)*
*func TestFeaturesPeggy(t *testing.T)*

The above three unit test functions test if the addition of the feature of personality for the matching quiz is correctly added to each cat in the shelters.

We also were able to fix the following unit tests:

*func TestGetCats(t *testing.T)*

This unit test creates a router and checks for a 200 ok status to see if it pings to the correct routing of getCats. The test also checks for an expected body response of getting all the cats.

*func TestGetCat(t *testing.T)*

This unit test creates a router and checks for a 200 ok status to see if it pings to the correct routing of getCat. The test checks for an expected body response of a singular cat.

*func TestUpdateCat(t *testing.T)*

This unit test creates a router to help test for the updateCat method and compares if it connects by testing the connection status. The test then checks for an updated cat JSON struct.

*func TestCreateCat(t *testing.T)*

This unit test creates a router to help test for the createCat method and compares if it connects by testing the connection status. The test then checks to see if the cat struct has been created.

*func TestDeleteCat(t *testing.T)*

The unit test creates a router to help test for the deleteCat method and compares if it connects by checking the connection status, the status should be NoContent. If the correct status shows up, then the test has passed.

We were able to fix them because in the previous sprint, we had placeholders that were not the actual extracted cats from the shelters to be able to test if the functions worked. However, now we were able to hook up the actual extracted cats and display them when the tests are being ran.