Test cases

1. Unit tests: Tests on individual components

I should first note that almost everything in this application is an integration (or regression) test on some level, because they all rely on a working display from the component in which the rendered object is embedded.

As a rule, the component is <ComponentName property1={data} … propertyN={data}/>

When I say to “hard code” a property, it’s inputting data in property1 … propertyN

**Component: App and Home pages**

Test Method: Add <div>text<div> in render.

Expected: Text displays

*All other tests involving App will be regression and integration tests by embedding other components.*

**Component: PetDisplay, PetCard, ProviderCard, UserProfile** (or whatever it’s called)

Test Method: Display the component by providing it hard-coded information that would appear in the pet’s display. For instance, hard-code the petName property in the calling function.

Expected: The hardcoded pet’s information displays correctly.

*This is the same test for all components that will ultimately display information saved in Firebase.*

**Component: PetList and Provider List**

Test Method: The filtering requires an array of pet (e.g.) objects, which will have to be provided. The filter will iterate through the provided array and check if each of those pet objects contains the necessary filtered property, e.g. “Gender” or “Animal Type”.

Expected: Only the objects containing the filtered information appear.

This is the same unit test for any other component that will ultimately display a list of objects retrieved from the database.

**Component: Login**

Test: Verify that the form to log in will toggle when the logged-in property is manually set to true or false.

Expected: The form appears when the property is set to false, and the button to log out appears when the property is set to true.

**Component: PaypalRender**

Test: The actual donation will be done as an integration test. This test is simply to verify the display mechanism of the forms and images.

Expected: Verify that the paypal button, dollar amount form, and dropdown menus appear

**Component: AddPet, AddPetProvider, Register**

Test: Fill in the form. Set the onSubmit method to display a success message.

Expected: The form retains the typed in data (so the onChange is working), and on Submit displays the success message.

*Repeat as an integration test with database tools, where onSubmit sends the information to the database tools library.*

**Component: NavBar**

Test: Add links to display

Expected: The links display and are clickable (even if they don’t go anywhere)

*Repeat as an integration/regression test whenever the linked-to component is added.*

2. Integration Tests: Tests between components

a. Repeat unit tests where indicated as integration tests with the database dynamically supplying info.

b. i. Test that the Donate function properly calls the payment processor. Expected: Payment object returned and stored as a transaction record.

ii. Test that “browsing” (simplistic search) properly calls the database as required. Expected: All pets get displayed.

iii. Test that searching properly filters with the information provided. Expected: Only pets with a particular filter are displayed.

c. Verify that the payment processor processes the payment. Expected: same as above, this is just the paypal side of the test.

d. Verify that the Database generates unique IDs for users, pets, and pet providers. Expected: Handled automatically by Firebase, just look at the real-time database and verify that the object was imported.

e. Re-verify 1c after the GUI is created, this time using the GUI functions to access the editing features. Expected: Same as 1c.

f. Verify that the pages display as intended. Expected: Just what it says on the box.

~~g. Verify that the payment system returns a receipt and information to be stored in the donation info for each pet provider.~~

h. Verify the user account creation functions. Expected: User account is actually created in the database.

3. Regression tests: tests to be completed after each substantial change to the system.

a. High priority: Re-verify 2b, 2e, 2f.

b. Moderate priority: Re-verify 2a, 2c, 2d, and 2g.

c. Low priority: Verify that the appearance of the site doesn’t change.

Unit tests should be repeated on any class that is changed before integration tests involving that class are repeated.

4. System tests: Tests as-if end user

a. Browse and search for pets. Expected: Pets are returned with the properties specified.

b. Donate to a provider or the site. Expected: Donation is completed to the provider specified. [This might not be possible due to the way PayPal handles developer sandboxes.]

c. Create a basic user profile and log in. Expected: Account is authenticated, and app registers the user as logged in.

d. Adopt a pet. Expected: Pet is removed from searchable results and appears in User’s pets.

e. View my pets. Expected: Pets for the user are displayed.

f. Alter the pet’s profile, including the medical records (if application). Expected: Pet’s information is altered.

g. Alter my profile. Expected: Same as f.

~~h. Donate.~~

i. Log out, then log in again, then delete my profile. Expected: Profile’s credentials don’t result in beign able to log in anymore.

~~j. Create a GlobalAdmin.~~

k. ~~Alter a user’s profile.~~ Alter a PetProvider’s information. Expected: The information in the pet provider account the user admins for is changed.

l. Delete a user’s profile. Delete a pet provider. Expected: the profile or account is deleted.

m. Add a pet provider and assign admin status for that PetProvider to a user account. Expected: The pet provider profile shows up and recognizes the user as its admin.

n. Log in as admin for pet provider. Expected: Able to perform the admin tasks on that pet provider.

o. Perform administrative actions on the Pet Provider page, and re-test all the regular user actions (d—i). Expected: See above.

Wishful thinking:

p. Create a Vet account. Re-do the regular user actions. Expected: See above.

q. Insert a pet’s medical records, then edit them. Expected: Medical records file is viewable and can be changed by vet.

r. View your patients, add a patient, delete a patient. Expected: patients (pets) are visible in this profile’s My Patients. New patient becomes visible when added. Patient stops being visible when deleted.