Test cases

1. Unit tests: Tests on individual components

a. Instantiate each object and verify that the object displays its correct information. If an object automatically adds information from a database, for the unit tests you may be required to provide the information manually. Meaning if the object pulls the pet name from the database and stores it in the variable, hardcode the variable until the database is hooked up. (This test should be completed as an integration test.)

Object-specific unit tests (if there’s anything really special in an object)

~~b. In User objects:~~

~~i. Test that the Donate function properly calls the payment processor.~~

~~ii. Test that “browsing” (simplistic search) properly calls the search plugin as indicated.~~

~~iii. Test that searching properly calls the search plugin with the information provided.~~

c. In all profiles, verify that the edit profile function works. Before the GUI is created, this should be simply verifying the getters and setters; after the GUI is created, this is an integration test.

d. For animal types, just verify that our language’s version of instanceof returns the correct type of animal. (For the most part, the animals will not have their own variables, but instead will simply be identified by the type of object they are.)

2. Integration Tests: Tests between components

a. Repeat 1a as an integration test with the database dynamically supplying info.

b. In User objects:

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 search plugin as indicated. Expected: All pets get displayed.

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

c. Verify that the payment processor and search plugins return the correct information. (E.g. when called by a user) Expected: same as in b, this is just the other 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.