SW Engineering CSC648/848

DoggyDex Web Application
Section 01, Team 02

Grayson Dew - Team Lead
Orion Culbertson - Front-end Lead
Miguel Galvan - Git Master
Rigo Perez - Back-end Lead
Kristopher Phillips - Scrum Master

Milestone 4
12/2/2021

History Table:

Version	Date
V1	12/2/2021

1. Product Summary

DoggyDex

DoggyDex is a new and exciting upcoming app for all dog lovers, dog enthusiasts, and even simply the dog-curious. It is easily accessible for all ages and backgrounds, with the purpose of assisting its user in identifying dog breeds they see around them, as well as facilitating an interesting learning experience when it comes to our canine companions.

Our product offers allow for a smooth and personalized experience by utilizing some key features: User Accounts, Creating an Account, Logging into an Account, Uploading a Photo, Personal DoggyDex, and Learning Mode DoggyDex. User Accounts are the first step, and an important aspect for a unique adventure tailored to the individual. After creating an account, the user may log in to view a list of all the dog breeds they have discovered. Discovering a breed is as easy as uploading a photo! Once uploaded, DoggyDex will identify the dog breed and unlock it for viewing in the owner's personal list of discovered dogs. For users less intrigued by finding all the dog breeds on their own, our app also features a toggle that will switch from a personal DoggyDex to a "Learning Mode". Enabling learning mode reveals all dog breeds, allowing all information about each to be instantly accessible.

DoggyDex stands out above its competitors in a myriad of ways. Immediately noticeable is how clean and visually appealing the interface is compared to other complex and cluttered apps. Our dog breed identification boasts a greater than 90% success rate, a nearly 20% increase over similar apps. Unlike other apps, the user's DoggyDex is revealed as photos are uploaded and identified, allowing for the personal collection feel we pride ourselves on.

There are several goals we have in mind for DoggyDex, and all of them rely on you. Between education, exploration, and more than anything else, entertainment, there is something everyone can enjoy when they use this app. We hope that you like it as much as we do.

- URL: http://doggydex.xyz/

2. QA Testing
    a. Unit Testing (for 3 major user stories)
        i. User Story 1: Create an Account
            1. Tested UI/Front-End: Filled out form by rendering React component Checking that all input fields were blank and followed that with filling out the form by firing events. Tested fields with expected values and tests passed. Registered new user by sending axios request to backend. Received a successful status code and message. Confirmed in the database that the user was registered correctly.
               Attempted to register a user with an already registered email. Attempt failed as expected.
            2. Testing tools: Javascript, Jest, React testing library, axios. 3/3 tests run successfully.
            3. Isolated CreateAccount and feature and assured major functions worked. In the future, we should test the confirm email and confirm password to make sure they're working as expected.

        ii. User Story 2: Add dog to personal DoggyDex
            1. Mocked a backend database and made mock api calls to add dogs to user's personal doggydex. Tested that every add the number of dogs in users list of found dog increased.
            2. Testing tools: Jest, React testing library, Supertest and Jest-mock for mocking backend and http calls.
            3. Should test with actual backend to ensure that such functionality works. Decided against testing with actual database because of the crossover Units: Register a guest or Login a guest.

        iii. User Story 3: Switching between Learning Mode and Personal DoggyDex
            1. Simple test that checked that checkbox could change state on click.
            2. Testing tools: Jest, React testing library, Jest-snapshots.
            3. Wanted to take snapshots of when a user was in Personal or Learning mode and compare the two. I was also researching on how to compare child component or a parent depending on state. The DoggyDexToggle Complement renders different childs(PersonalDoggyDex or LearningModeDoggyDex) depending if the user clicks in and out of the learning mode or personal mode.
    b. Integration Test (for all P1 features) - Orion
        i. Create Account

Test Case ID	1

Test Case Description	Create new DoggyDex account with Test Data information
Date Tested	11/24/21
Test Scenario (w/ steps)	1. Navigate to http://doggydex.xyz/create-account a. Expected Result: Blank Create Account Form loads 2. Enter Test Data into the Create Account form a. Expected Result: Form is able to be filled 3. Click Create Account button a. Expected Result: Redirects on Success
Prerequisites	Access to Chrome and Firefox browsers
Test Data	● Name: FireFox User, Chrome User ● Username: fireUser, chromeUser ● Email: fire@dog.com, chrome@dog.com ● Verify Email: fire@dog.com, chrome@dog.com ● Password: dogPassword1 ● Verify Password: dogPassword1
Test Results	Browser: Chrome Result: Pass except for the redirect on success Browser: Firefox Result: Pass except for the redirect on success

ii. Login

Test Case ID	2
Test Case Description	Log In with Test Data information
Date Tested	11/24/21
Test Scenario (w/ steps)	1. Navigate to http://doggydex.xyz/log-in a. Expected Result: Login Page loads 2. Fill in Login Form with Test Data a. Expected Result: Login form is fillable 3. Click Login Button a. Expected Result: User gets logged in and redirected to their User Profile
Prerequisites	Access to Chrome and Firefox browsers
Test Data	● Email: fire@dog.com, chrome@dog.com ● Password: dogPassword1
Test Results	Browser: Chrome

	Result: Pass Browser: Firefox Result: Pass

iii. Photo Upload/Dog Breed Gets Classified

Test Case ID	3
Test Case Description	A photo of a dog gets uploaded and the ML returns a result.
Date Tested	12/2/21
Test Scenario (w/ steps)	1. Download the image provided in the Test Data as a .jpg file. 2. Navigate to http://doggydex.xyz/ 3. Click "Select Dog Photo…" 4. Select the image provided. 5. Click Submit 6. Wait for Response a. Expected Result: German Shepherd result gets displayed with confidence score.
Prerequisites	Access to Chrome and Firefox browsers
Test Data	This image
Test Results	Browser: Chrome Result: Fail Browser: Firefox Result: Fail - The ML wasn't quite able to get up and running yet.

iv. Personal DoggyDex

Test Case ID	4
Test Case Description	Dogs displayed accurately
Date Tested	11/29/21
Test Scenario (w/ steps)	1. Log in to http://doggydex.xyz/log-in with information from Test Case 2 a. Successfully logs in 2. Navigate to http://doggydex.xyz/doggydex a. Learning Mode DoggyDex loads 3. Select "Switch to Personal Mode" at bottom of the page

	a. Toggles to Personal DoggyDex with found dogs loaded. (2 should be selectable)
Prerequisites	Access to Chrome and Firefox browsers
Test Data	N/A
Test Results	Browser: Chrome Result: Pass Browser: Firefox Result: Pass

v. Learning Mode DoggyDex

Test Case ID	5
Test Case Description	Making sure the Learning Mode DoggyDex navigable and that the DoggyDex entries can be accessed.
Date Tested	11/24/21
Test Scenario (w/ steps)	1. Navigate to http://doggydex.xyz/doggydex a. List of available dog breeds gets shown 2. Click on one of the available dog breeds a. Dog Breed Info page comes up 3. Click back button on bottom of page a. List of available dog breeds gets shown again
Prerequisites	Access to Chrome and Firefox browsers
Test Data	N/A
Test Results	Browser: Chrome Result: Pass Browser: Firefox Result: Pass

vi. Analysis of your test coverage for your P1 features and describe it.

1. In order to comply with the page limit restriction for our testing section, we were only able to conduct one integration test for each of our P1 features. This gave us about 33% test coverage for our P1 features.

3. Code Reviews

a. Coding Style Report:

        i. The way we decided to approach coding style was to all agree on an IDE and a style extension to all commit to using for this project. We all agreed to using Visual Studio Code and the Prettier extension. For most settings, we agreed to use the default settings provided in the Prettier module, but we did encounter some inconsistencies when dealing with quote marks where single or double quotes were used. We ended up deciding to use single quotes in all cases where double quotes were not mandated syntactically. As we did not start this project using an agreed upon coding style, we have since gone through the files throughout our project and applied the Prettier formatter to them as we have been working on them when implementing new features or debugging. This approach covered most documents in our project, but an additional project-wide sweep was still necessary to ensure compliance.

    b. GitHub Pull Requests showing Code Review Practices

        i. https://github.com/CSC-648-SFSU/csc648-fa21-01-team02/pull/43

        ii. https://github.com/CSC-648-SFSU/csc648-fa21-01-team02/pull/45

        iii. Our code review process has been a highly collaborative one. Throughout the implementation phase of this project, we have used the time in our team meetings both as the regular scrum meetings to discuss task delegation and implementation strategy and also as opportunities for us to vocalize any issues to the entire team. This code review strategy became less effective as time went on because the project grew in size and each team member specialized more and more, but we still largely achieved our goal with this whole-team approach which was to utilize each other as resources to learn the various technologies we were using for our application. This was our goal from the outset because every team member was exposed to at least one new part of the technology stack, so we all had something to learn from this project. That being said, later on in the implementation of the project, we began relying on code review from just 1 other team member when a pull request from an individual's feature branch into one of our permanent branches was made. This prevented the team from getting unnecessarily bogged down in our limited time during team meetings. Above are links to examples of 1 other team member providing feedback on pull requests from a different team member's pull request.

4. Self-check

Application will initialize within 3 seconds after showing the title screen.	DONE

Password requirements with at least one of: Upper Case, Lower Case, Number	ON TRACK
Application should be able to have a user capacity of at least 1000 before failure.	DONE
Dog breed identification should take less than 3 seconds.	ON TRACK (currently takes about 15 seconds)
The Home screen (first screen the users will see) will have an attractive layout with buttons for features. Navigation will be natural and easy for users to understand.	DONE
Transition between features should be 0.1 seconds or less.	DONE
Data will all be hosted on Amazon Web Services.	DONE
All code will be maintained through the collaborative use of GitHub.	DONE