# CS373: Software Engineering

# 10am, Group 3 - Pets4me

Connor Sheehan, Rosemary Fortanely, Cristian Garza,
Robert Hrusecky, Andrew Cramer, Dean Torkelson

# Section 1: Motivation

The motivation behind Pets4Me was the lack of a single aggregator for pets that are up for adoption in the Austin area. There is never a shortage of animals that need homes and families, but finding the perfect pet requires going to many different shelter websites, visiting many in person, or spending time calling multiple sources for information. However, with Pets4Me, all the data a future owner needs (be it on the shelters, different breeds, or the animals themselves) is presented in one single location, making it easier for animals in need to find their forever home.

# Section 2: User Stories

## User Stories for us:

1. As a new user, I want to be able to browse the most popular breeds for adoption at specific shelters so I could learn more about the local adoption trend.
    a. This feature was not implemented because our data is aggregated from the currently available pets, and not previously available/adopted pets.
2. As a new user, I want to see brief introductions on how to take care of specific breeds so I can take action to help the animals.
    a. This feature was not implemented because our APIs used for cats and dogs do not provide information on breed-specific care.
3. As someone interested in a specific breed, I want to be able to see the changes in adoption statistics for that breed so I can learn if the situation is getting better for it or not.
    a. This feature was not implemented because our data does not include information on adopted pets and we cannot then provide insights on data pertinent to adopted pets.
4. As a user, I want to know if the shelter closest to my place has a pet I am interested in so I can plan where to go to adopt the pet.
    a. This feature was not implemented because searching for shelters by location is not a requirement for Phase 1 and will be implemented later. Since the API provides functionality for shelter distance by location, we will have to provide a search box for the user to enter a zipcode or city, state combination, and then retrieving the location will be as simple as making an API call to the PetFinder API. Then the cards will need to be sorted in ascending order so the user will first see the closest shelters. It is estimated this task will take 2 days, one for making the API call to

PetFinder for shelter distance, and one for implementing a JavaScript function to sort the cards.

5. As a pet lover, I want to be able to know which two breeds go well together so I can decide on my second pet.
    a. This feature was not implemented because our APIs used for cats and dogs do not provide information on which breeds get along best.

### Our User Stories for CrashSafe:

1. Responsive Home Page: As a user, I want to be able to resize the home page so that I can consume a responsive website. Currently, the website does not render the image and 'Enter' button correctly for the desktop and mobile view. The ease of use of the website would be improved if I could navigate throughout the home page without the image being incorrectly displayed or the 'Enter' button being truncated and shifted off to the right of the page.
2. Enter Button Functionality: As a user, I would like to be taken to somewhere when I click the 'Enter' button on the home page of the website. One possible function it could perform is it could take you to the brands' page. Another idea might be to create buttons for each model and display basic statistics about the models.
3. Developer Bios: As a user, I would like the about page to contain biographical information for all the developers of this project. I think these are important and will help customers like me feel more connected to you. These biographies do not have to be lengthy, but they should help us get a rough feel for the kind of people we are working with.
4. DNS redirect: As a user, I would like to be able to visit the website at crashsafe.me so that I can access the stable build more easily. Currently, the only way to view the website is to run it and look at localhost. It would be much easier to suggest features if I could access the latest stable version of the website in my browser instead. Hopefully, it is hosted on some slick platform like AWS or GCP.
5. Add instances to model homepages: As a user, I would like to see different car brands under the "Car brand" page. I would also like to be able to search and sort these. Lastly, I would like this feature to be user-friendly.

## Section 3: RESTful API

### GET /models/pets

This endpoint will return all instances of individual pets.

### GET /models/breeds

This endpoint will return all instances of breeds.

### GET /models/shelters

This endpoint will return all instances of shelters.

### GET /models/pets/{id}

This endpoint will return the pet that has the id specified by the parameter.

### GET /models/breeds/{id}

This endpoint will return the breed that has the id specified by the parameter.

### GET /models/shelters/{id}

This endpoint will return the shelter that has the id specified by the parameter.

## Section 4: Models

For our three models, we decided on 1) the individual pets up for adoption, 2) the different breeds of dogs and cats, and 3) the shelters where you could adopt the animals.

### Model 1: Pets

The pets model contains the “dog-ographic” information on the individual animal. We included information about the pet itself, such as name, breed, size, gender, and age. Each pet model also contains information about the adoption, such as where the animal currently is, how much it costs to adopt, and a link to the shelter’s website. From a pet’s page, you can be linked to the page for the breed, as well as the page on our website for the shelter.

### Model 2: Breeds

The breeds model contains the information that someone who has never heard of it would need to get an idea of what the breed is like, such as its name and nicknames, species, origin, average lifespan, size, typical traits, and how heavily it sheds. From each breed page, you’ll be shown a few links to animals that are up for adoption that match that breed, as well as the shelter that has the most animals of that breed. However, for Phase 1, we haven’t yet implemented those links.

### Model 3: Shelters

The shelter model has essentially every piece of relevant information that someone looking to adopt would need. Including its location, available hours, contact information like emails and phone numbers, adoption policy and mission statements, and of course their social media. If there is any information that someone is looking for that we don't have, there is a link to the shelter's website on the page.

## Section 5: Tools

The primary tools we used in this first phase were React, TypeScript, Marvel, Postman, and Bash. Within React, we also used react-bootstrap and Material UI, as these both have a similar look and feel and allow us to easily implement complicated components. We chose to work in TypeScript as opposed to JavaScript in order to reduce the likelihood of type errors and silly human mistakes. We used Marvel to create mockups of the frontend for us to model our designs after. Postman was used to design our Phase 1 API. Lastly, Bash was used to deploy our website.

## Section 6: Hosting

We decided to host our website through GCP. We had the option of hosting through the Compute Engine and the App Engine of which we chose the latter. Although it was more difficult to set up (in our opinion, as with compute engine all that was needed was to start the server process in the VM), App Engine is free as long as your app stays within quotas set by Google. Also, App Engine provides more scalability features and it is easier to re-deploy once the project is set up. We got our domain name through NameCheap. HTTPS certificates are managed automatically through GCP App Engine by a feature known as "Google-managed SSL." This is advantageous because we no longer needed to know how to set up HTTPS or renew certificates manually. We found that the frontend instance needs the API key for our GitLab repository to aggregate statistics on the about page. We wanted to avoid storing the key in plaintext in our repository, as this would allow people without write access to our repository to make changes. To keep access to our repository secure, we opted to provide the key in an environment variable. However, the app.yaml file which provides a list of environment variables for deployment on GCP is also in our repository, so we could not store the key there either. Instead, we created a bash script that wraps the deployment process and allows you to specify the API key on the command line, or it grabs the environment variable from your development environment. It stores this temporarily in app.yaml for deployment and then restores the file from a backup after deployment is complete.