**Petfinder Documentation**

# **Introduction**

Petfinder is a crowd-sourced website for the users to search for a missing pet or report a missing pet. We used the combination of ReactJS and ReduxJS framework on frontend; used Spring Boot framework on backend that supports RESTful web service; and used MySQL as our database.

# 

# **Project Organization**

Project Members:

* Meng Wang
* Yixuan Liao
* Rui Chao/chao.r@northeastern.edu

# 

# **Requirements**

1. **Business Requirements**

This web application is intending to solve objective problems facing pet owners and stray animals.

* **For the pet owner’s side**: Efficiency for looking for missing pets, efficiency in adopting an animal, and reporting a missing pet.
* **For the stray animals**: Find a lovely home and get connected to the owner.

1. **User Requirements**

Petfinder provides user management functionality that allows users to register, login, logout, and visit their profile, and search/report missing animals. The navigation adapts depending on whether a user is logged in or not, and depending on their role in the application. There are two types of users or roles. A user can be a regular user or an admin.

The user interface adapts to the user role by hiding certain links, or entire pages and functionality. For instance, The Admin will have access to the administration page that would otherwise be inaccessible to the rest of the users of the application. The use cases for the two types of users are different to warrant different user interfaces. However, the login, register, and some profile content will stay the same interface.

1. **Functional Requirements**

* Home-Page (Web application presenting)
* SignUp-Page(Register as user)
* Login-Page(Access in personal information)
* Profile-Page(View and modify information)
* Search-Page(Function implemented for search usage)
* SearchResult-Page(Function implemented for search usage)
* PostTable-Page(Function implemented for user edit/update/delete/create)

1. **Implementation Requirements**

# Implementation: ReactJS,ReduxJS, SpringBoot, JavaScript, Java,HTML/CSS,MySQL

**HOMEPAGE:**

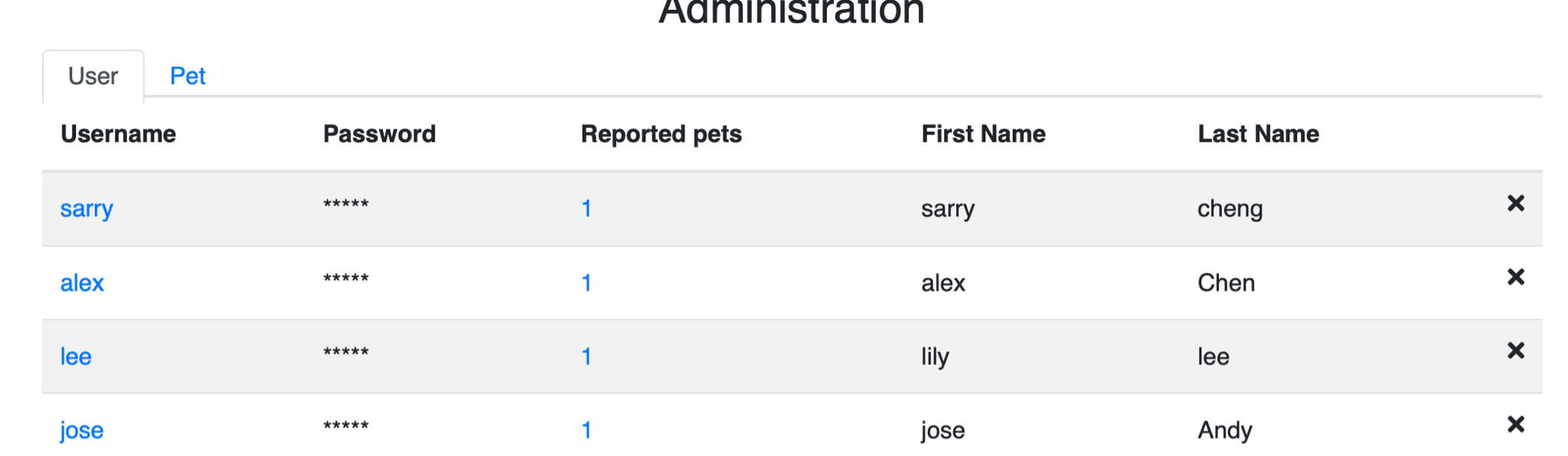
<https://petfinderadmin.herokuapp.com/>

Remote server for pet: <https://petfinderserver.herokuapp.com/api/pets>

Remote server for users:<https://petfinderserver.herokuapp.com/api/users>

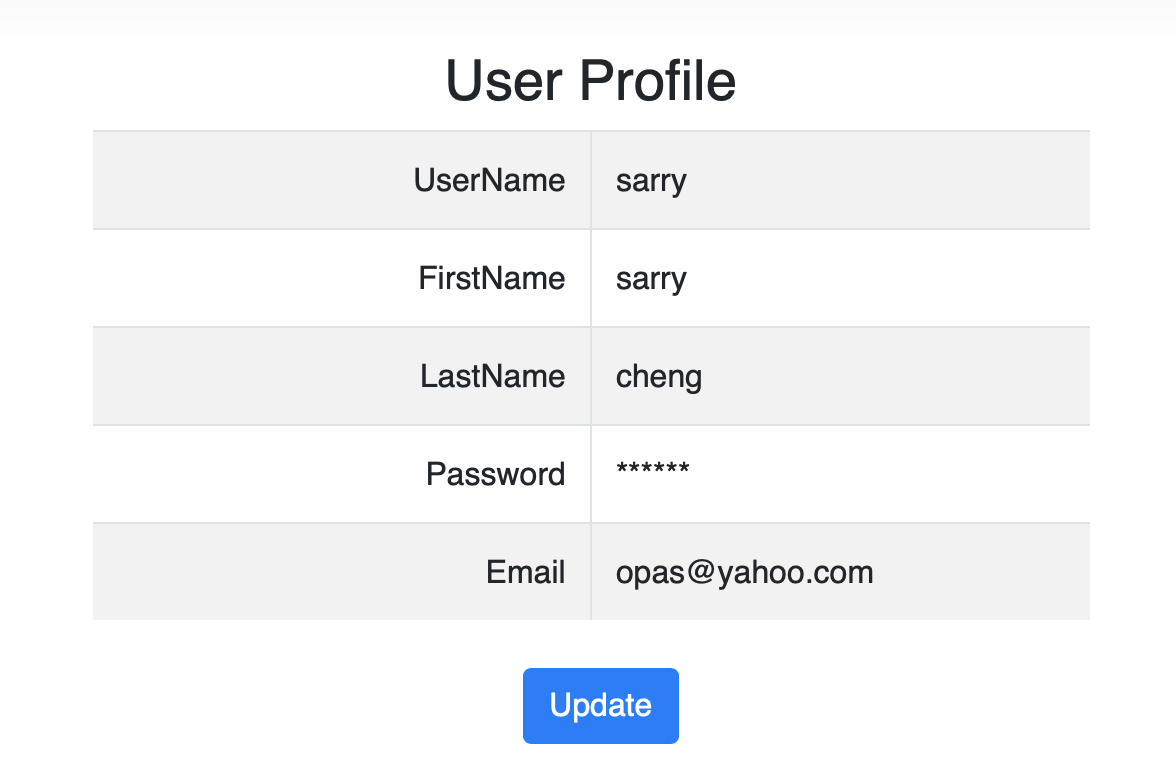
**ADMIN PAGE:**

1. Administrators can view users' lists.



1.1 When clicking on one user’s name, it will show the profile of the user.

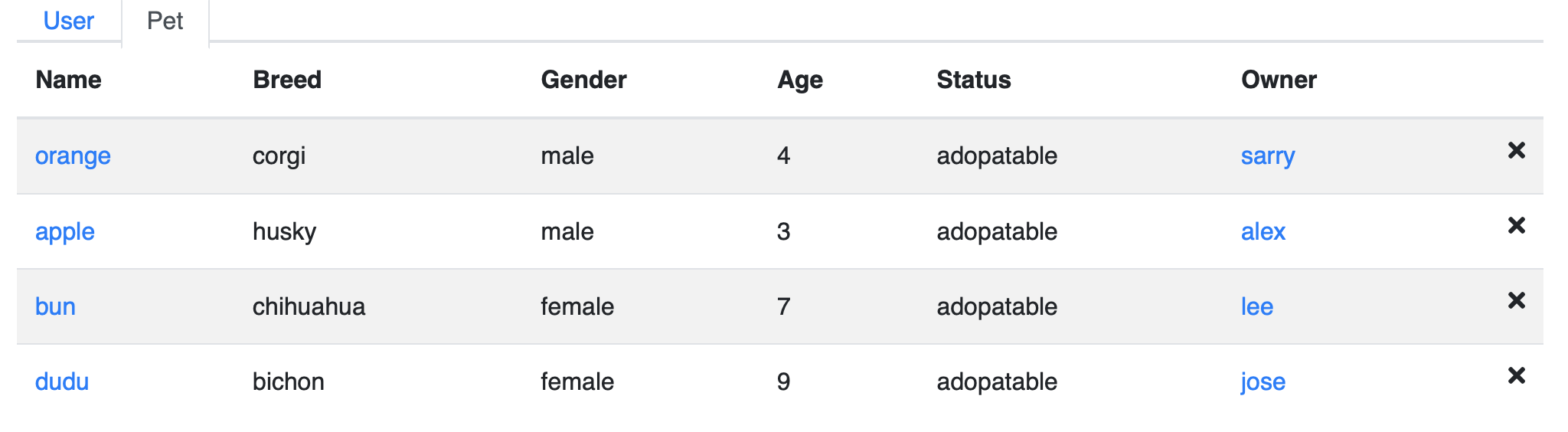
url example:<https://petfinderadmin.herokuapp.com/users/1234>



1.2 when clicking on the number of reported pets, it will show details of all the reported pets.

url example:<https://petfinderadmin.herokuapp.com/users/1234/pets>

1. Administrators can view the pets list.<https://petfinderadmin.herokuapp.com/admin/pets>



2.1 When you click on the pet's name, it will show detail-information about the pet.

url example:<https://petfinderadmin.herokuapp.com/users/1234/pets/234>

2.2 When you click on the owner's name, it will show detail-information about the owner.

url example:<https://petfinderadmin.herokuapp.com/users/1234>

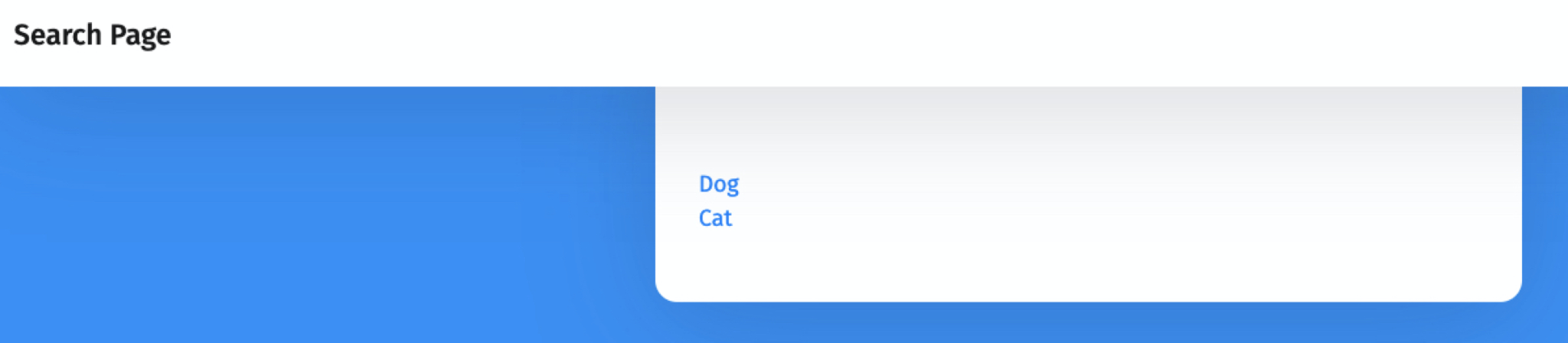
## **SEARCH PAGE:**

Search home page allows user to select either dog or cat:

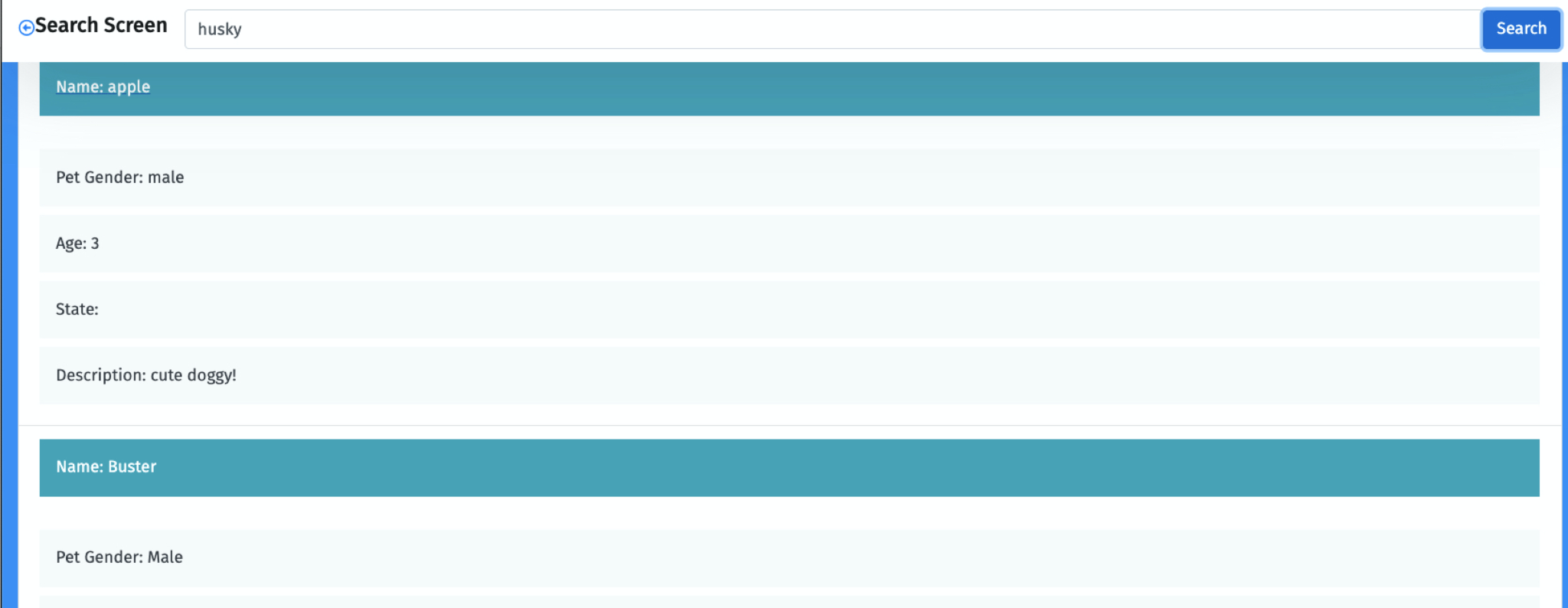
<https://petfindersearch.herokuapp.com/search>

Once user select it, the search box allows user to type any breeds for this type of pet,

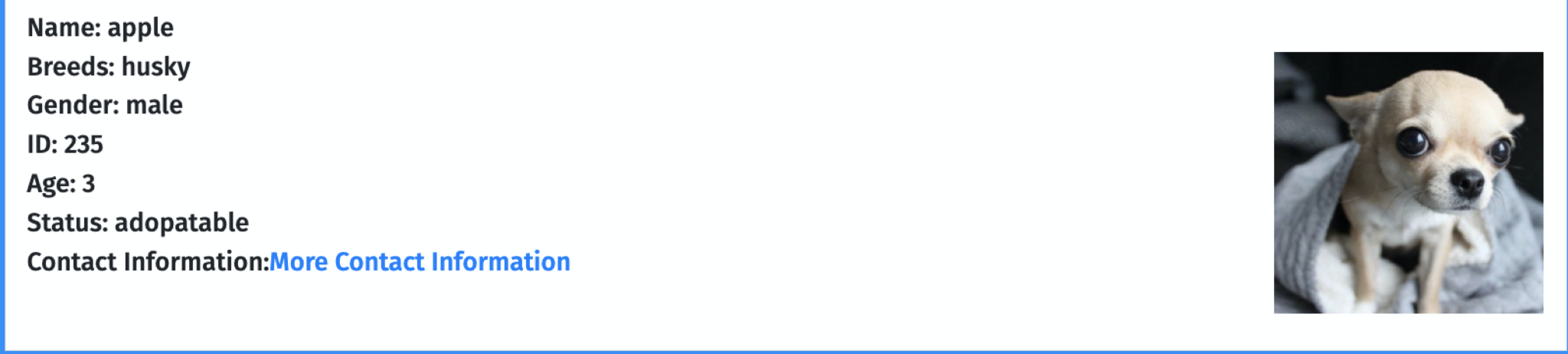
<https://petfindersearch.herokuapp.com/search/types/dog>



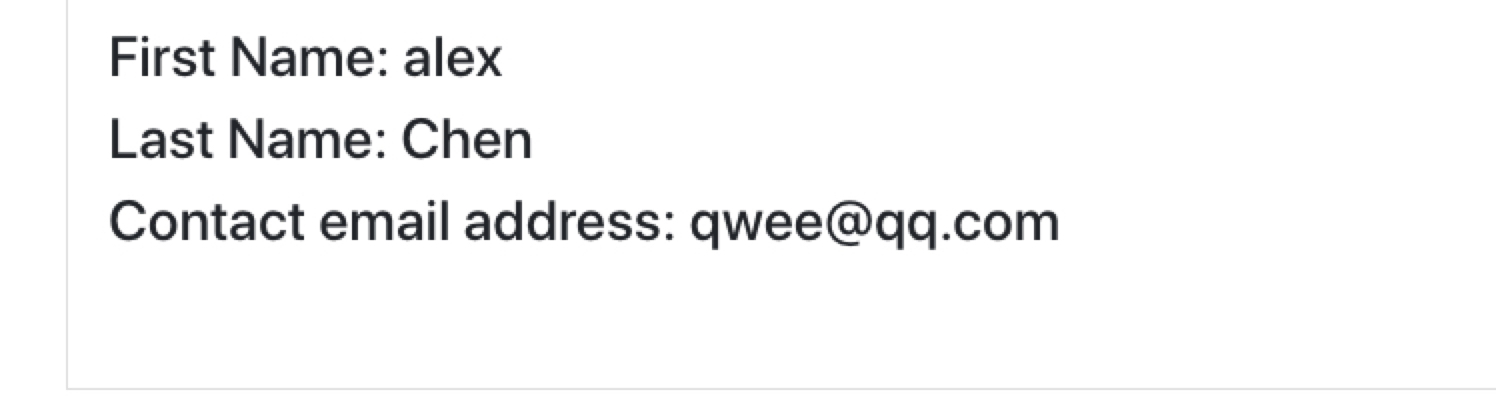
The pet’s information which was created by users should appear at the top of this page, and third party API data will run after that. For example, from this keyword “husky”, “apple” is from our remote server, and “D1” is from third party API

* 

Once user click on the pet information which is from our remote server, it allows user to click “more contact information”,



It could show more contact information of this poster,



## 

## **LOGIN PAGE:**

## 

**You can use our current testing ID (username: sarry password: 123), or use any username and password from** [**remote server**](https://petfinderserver.herokuapp.com/api/users)

1. Handle 5 Cases when user login

Case 1: when user not typing anything in both username and password,

Return alert message “Password is required” and refresh the login page.

Case 2: when user only typing username,

Return alert message “Password is required” and refresh the login page.

Case 3: when user only typing password,

Return alert message “Password is required” and refresh the login page.

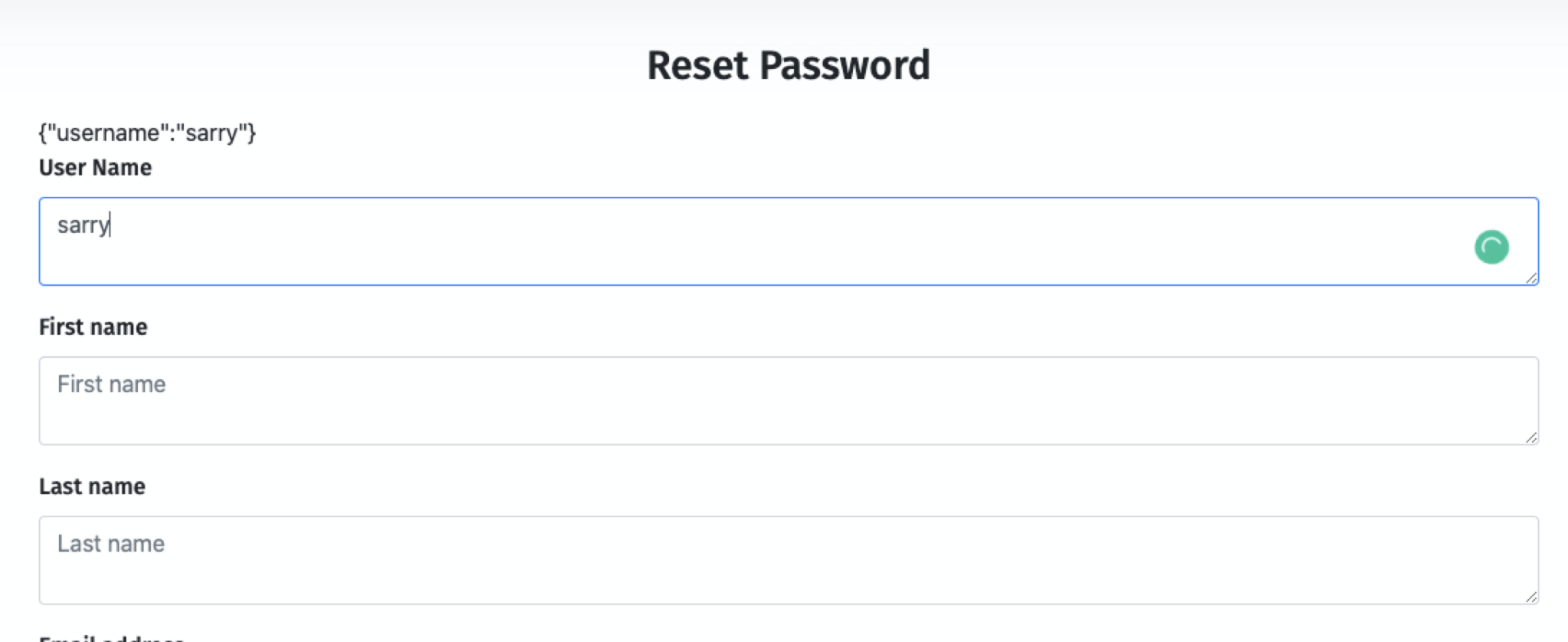
Case 4: when user type wrong password or wrong user name,

Return alert message “Password is required” and refresh the login page.

Case 5: when user type correct username and password,

Login Successfully, Return alert message “Password is correct” current page will direct to user’s Profile page.

2. Forget Password will direct to <https://petfinderlogin.herokuapp.com/reset>, user needs to remember his/her username and then to change other information, for example, the username is “sarry”, user can change his/her firstName, lastName or password.



## **SIGNUP PAGE:**

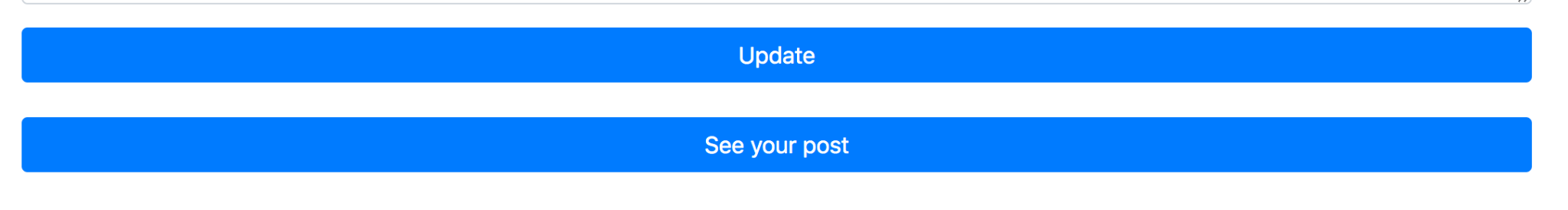
**Please ensure all information that is required is filled when you create a new user.**

We haven’t connected this application with the database yet, our current data is stored in an array, so if you leave content in blank, it will affect the testing process, since it will store the “null” data in our server.

1. Register allows choosing a role(s) for a user/admin.
2. Register allows users to register and create a new account
3. Register allows login in and identifying themselves
4. Register will be mapped to /login

## **PROFILE PAGE:**

1. Profile Page allows users to view their personal information. Users can directly change their personal information in provided space. Once the user finishes editing, the user can click the update-button to update information.
2. If the user has nothing to edit in the profile page, click “See your post” button will redirect user to the detail page(Post Table).



## **DETAIL PAGE(POST TABLE):**

This page can be viewed only by registered user, please use our current testing ID (**username**: sarry **password**: 123)

Once you logged in and entered the Post Table Page, you will see as below:

## 

Implemented: Create/Delete/Edit/Update functions in this page

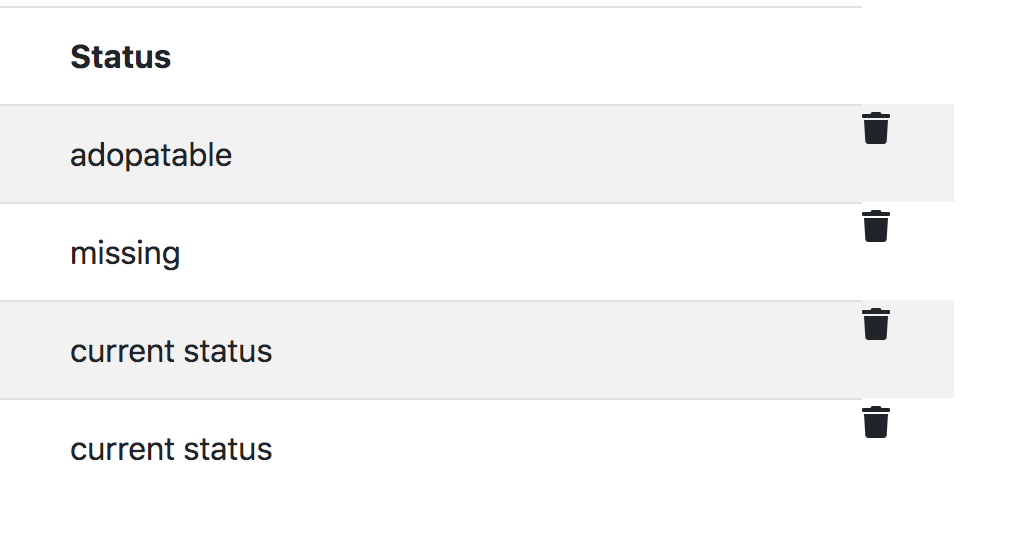
1. The “trash” icon on the left side allows the user to delete his/her post.
2. The “plus” icon allows the user to add a new post, once the user clicks it, the table will appear as a default post named “new pet”.
3. When users click “new pet” (redirect to a new page ) , if the user wants to edit information about this pet, click the “edit” icon, then he can edit the content provided below.

As the user is editing his content the “edit” icon will switch to “check” update-icon, Once he finishes editing, clicking update will store new information in our service api. (Image shows on below)

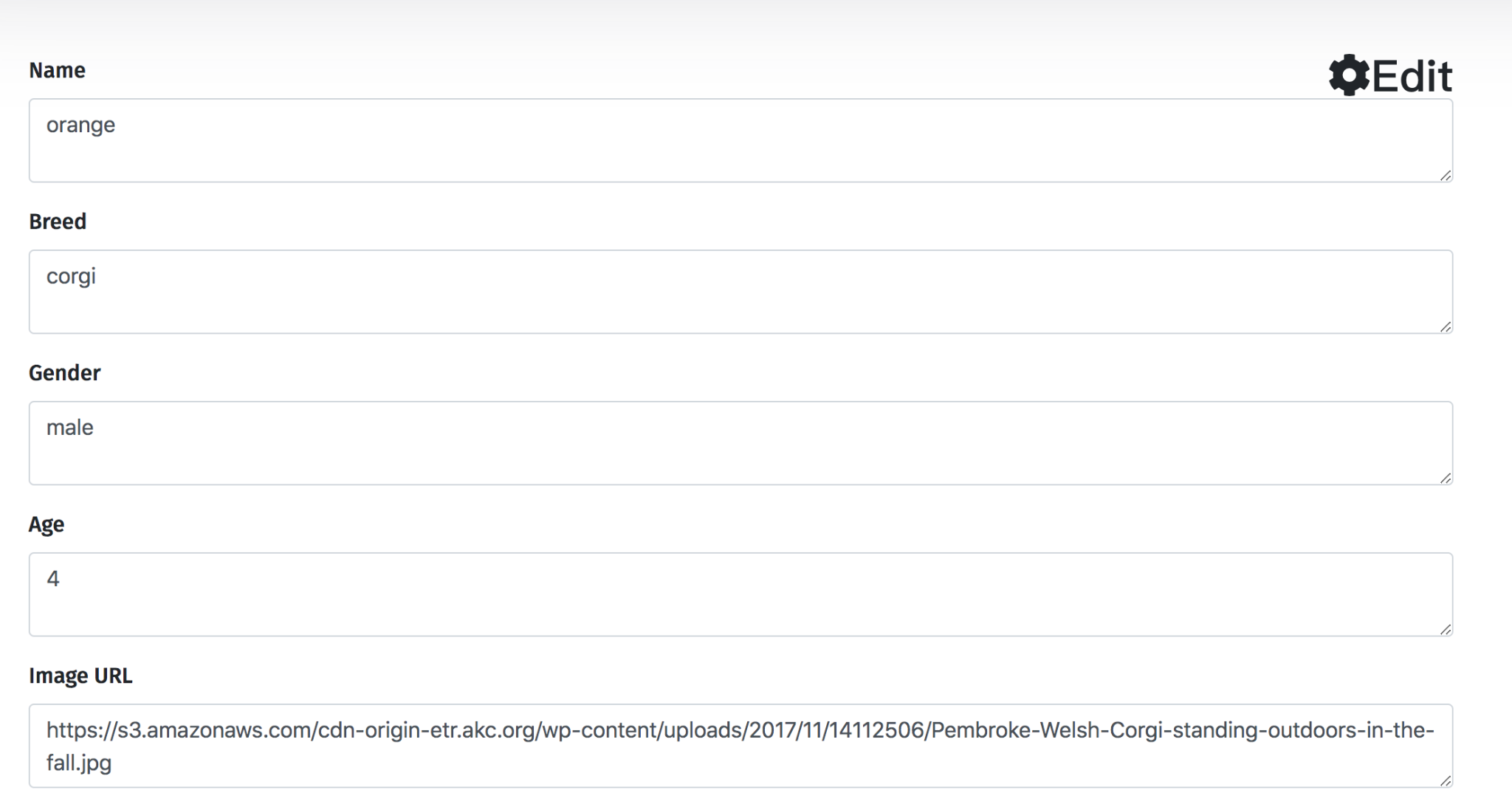
Add



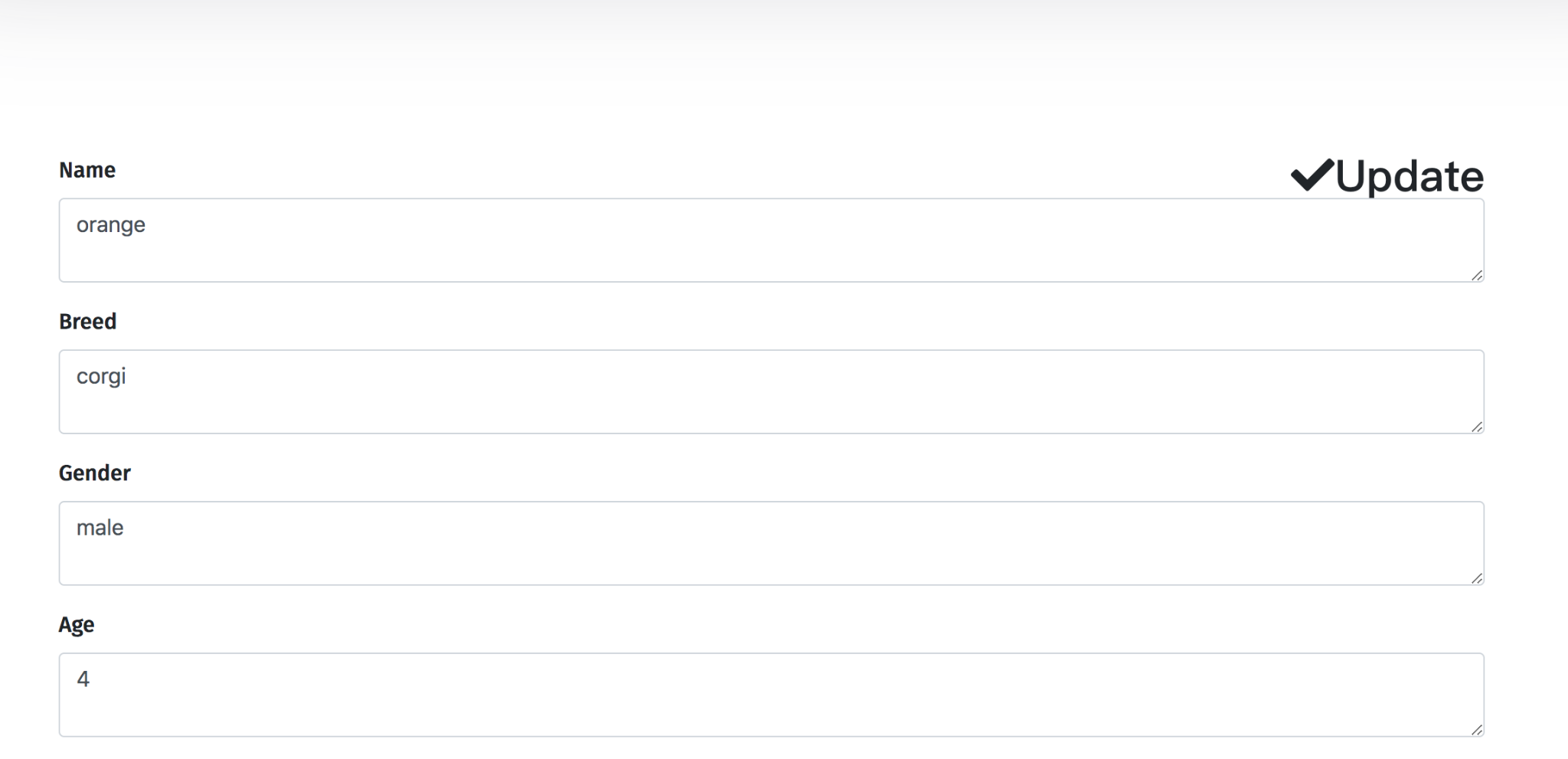
Delete



Edit(Please ensure to click “Edit” to edit any fields)

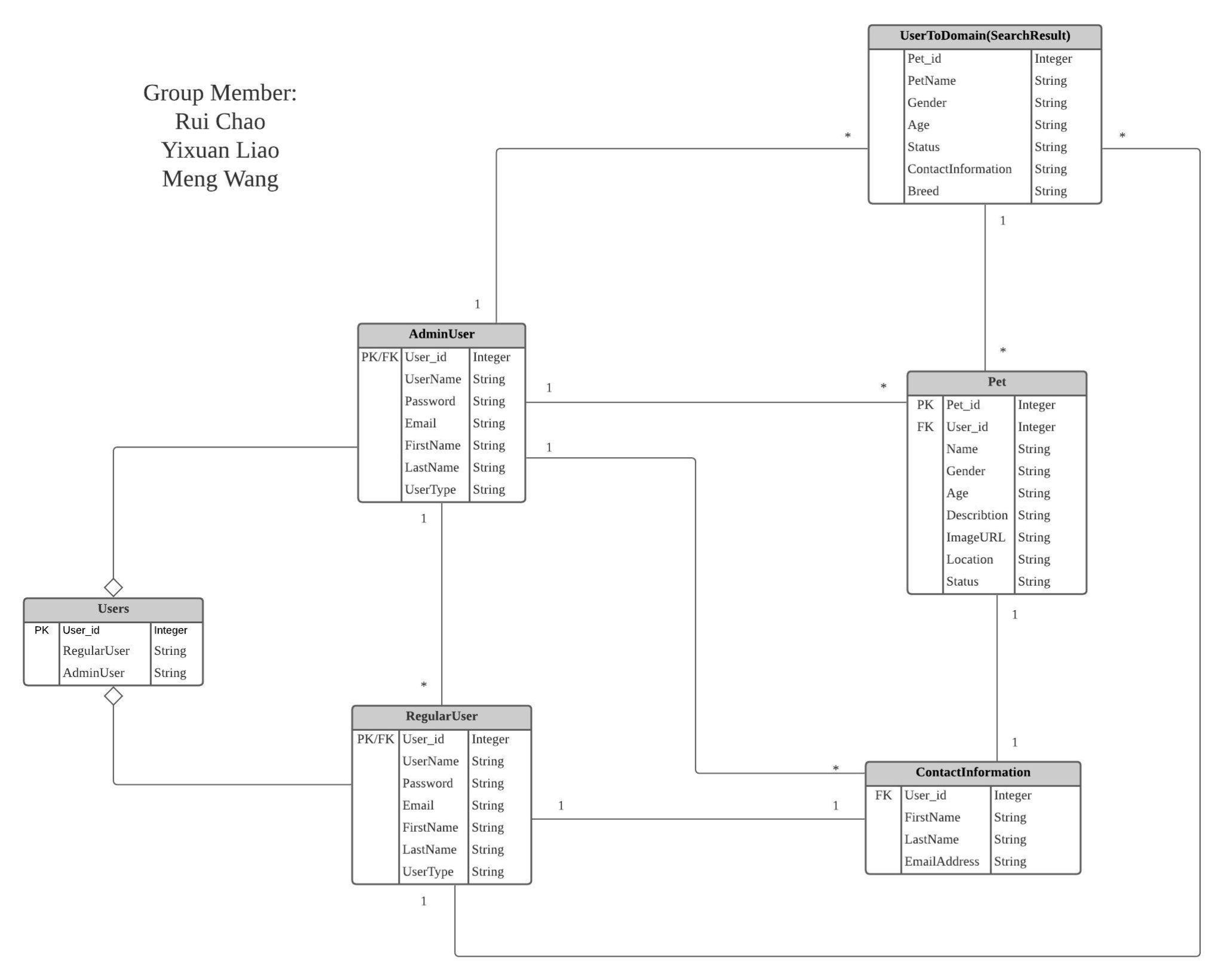


Update(Once update it, please click the “Update” button to show new fields)



## **Data Model:**

The diagram below shows the relation between several entities implemented in this project. Entities describe related attributes of an object or concept that we wish to store in a database. The User class contains attributes such as id (which uniquely identifies a User’s record), username, password, firstName, lastName, role, email, and phone. Admin and RegularUser classes inherit attributes from User, it is a more general form, or a generalization of Admin and RegularUser. Admin and RegularUser are special cases of a User. UserToDomain(Search Result) could be accessed by RegularUser and can be managed by Admin. Relationship of Admin and Pet is one to many.



## 