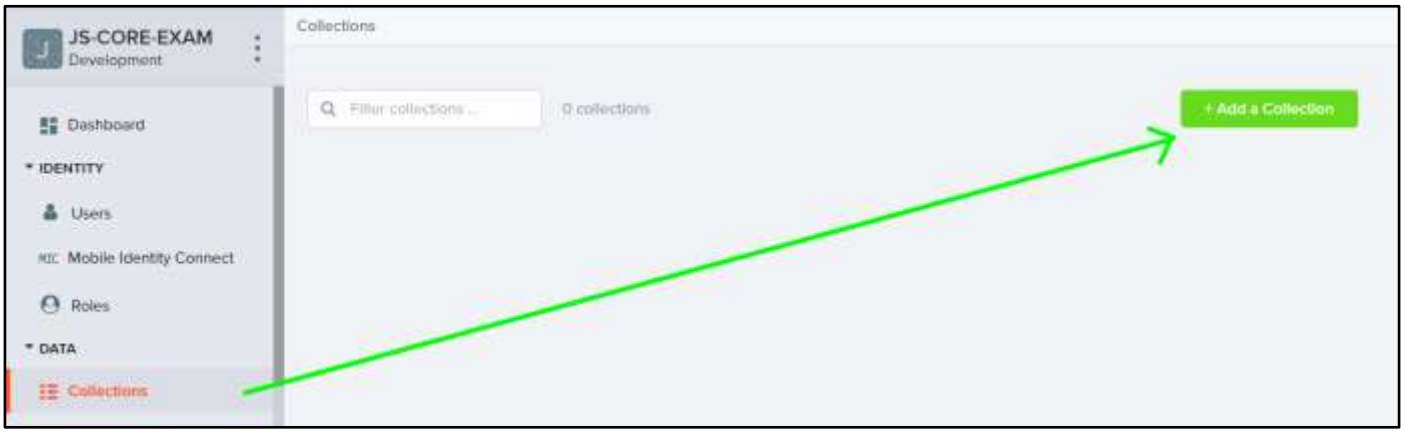
JS Apps Exam – Pet My Pet Single Page Application

Using libraries like **jQuery**, **Handlebars** and **Sammy** is allowed but is not obligatory. You are assigned to implement a **Web application** (SPA) using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend (Kinvey). The app keeps **users** and **pets**. Users can **register**, **login**, **logout** , view a **dashboard** with all **pets** sorted by **likes** (a helper function will be given). They will also be able to view the pets by **category**, **add** a new pet, **edit/delete** their **own** pets, show other pets **details** and a **section** where they can view their **own pets**.

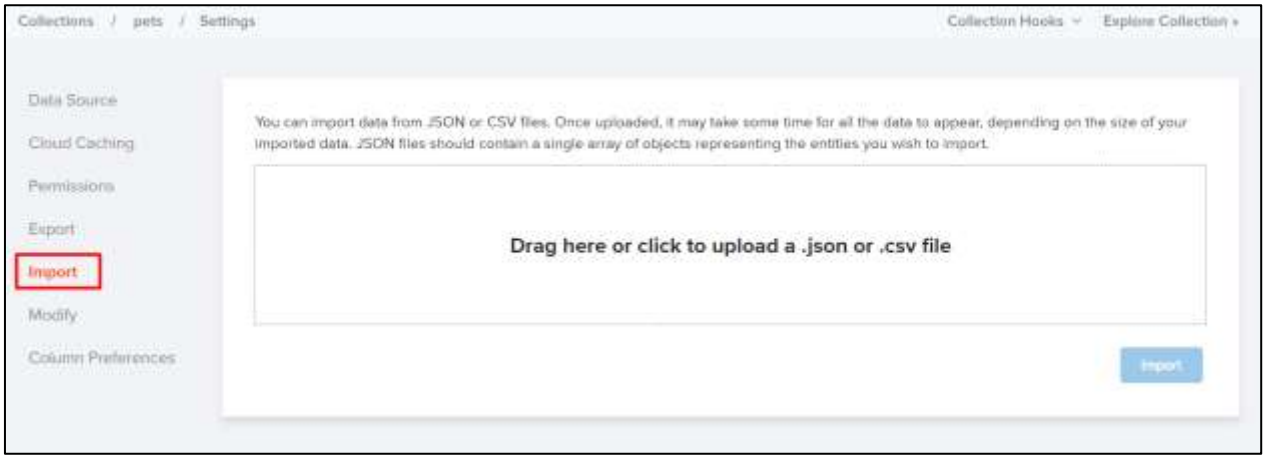
1. Create a Kinvey REST Service

Register at **Kinvey.com** and create an application to keep your data in the cloud.

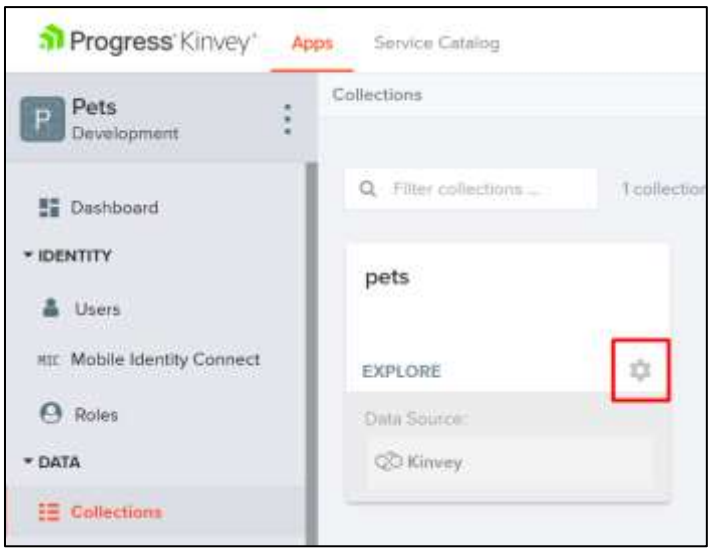
Create a collection **pets**. Each **pet** has a **name, description, imageURL, category** and an **likes**.



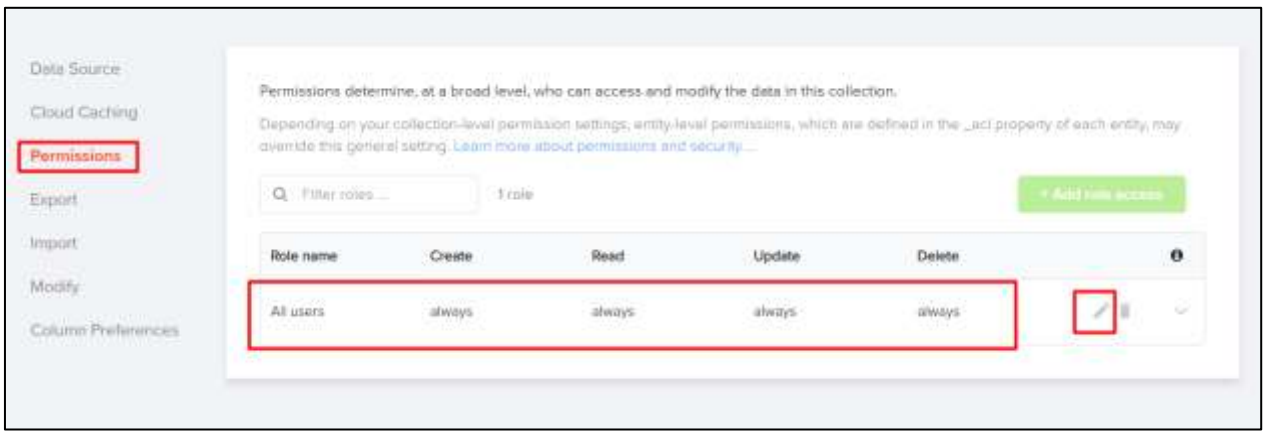
You are given **one** json file to **import** data for **pets.**



In order to be able to keep track of the **likes** of each pet, you need to **allow all users to edit this collection**. So go to the **properties** of the collection:



Then go to **permissions** and edit them to look like this:



2. Test the Kinvey REST Services

Using **Postman** or other HTTP client tool (you can use Kinvey's built-in **API Console**), test the REST service endpoints:

User Registration (Sign Up)

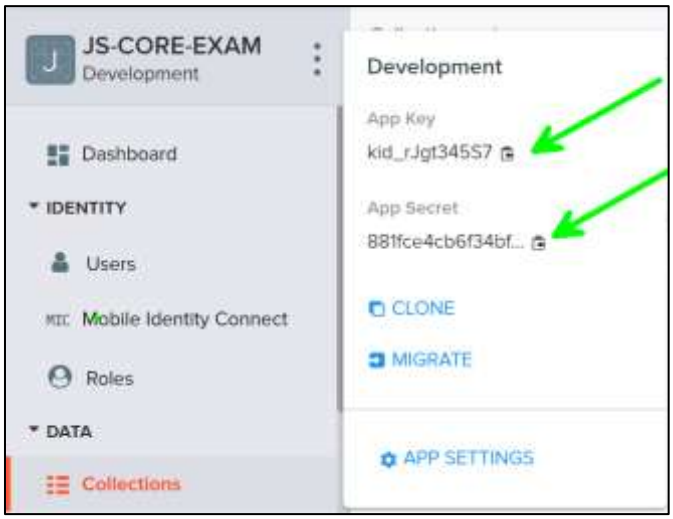
POST https://baas.kinvey.com/user/***app_id***/	
Request headers	Authorization: Basic base64(app_id:app_secret) Content-Type: application/json

Request body	{
	"username": "testuser",
	"password": "testuserpass890"
	}

Response 201 Created	{ "_id": "59930c78a743e20c7d3fca77", "username": "testuser",

	"password": "testuserpass890" }
Error response 409 Conflict	{ "error": "UserAlreadyExists", "description": "This username is already taken. Please retry your request with a different username", "debug": "" }
Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

The request needs "**Basic**" authentication. Use the Kinvey **App Key** and Kinvey **App Secret** as credentials.



User Login

POST https://baas.kinvey.com/user/***app_id***/login	
Request headers	Authorization: Basic base64(app_id:app_secret) Content-Type: application/json

Request body	{
	"username": "testuser",
	"password": "testuserpass890"
	}

Response
200 OK
	{
	"_id": "59930c78a743e20c7d3fca77",
	"username": "testuser"
	"_kmd": {
	"**authtoken**": "**8e6471bc-3712-4cfb-b92e-50e62a0c80....Duj5fHdM /7XHIe6KdY=**"
	...
	},
	...

	... }
Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

Successful login returns an "**authtoken**" which is later used to authenticate the CRUD operations.

User Logout

POST https://baas.kinvey.com/user/***app_id***/_logout	
Request headers	Authorization: Kinvey **authtoken**
Response 204 No Content	
Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

To logout, you need to provide the "**authtoken**" given by login / register as "**Kinvey**" authorization header.

List all Pets (Dashboard – sorted by likes, descending)

GET https://baas.kinvey.com/appdata/***app_id***/pets?query={}&sort={"likes": -1}	
Request headers	Authorization: Kinvey authtoken

Response
200 OK
	[
	{
	"name": "Lilly",
	"description":"This is my cat Lilly. She is 2 years old and is lazy",
	"imageURL":"https://www.bluecross.org.uk/sites/....jpg",
	"category":"Cat",
	"likes":"7"
	"_acl": {
	"creator": "5bb08c01eb7615589b4f360e"
	},
	"_kmd": {
	"lmt": "2018-10-03T08:04:26.028Z",
	"ect": "2018-09-30T09:26:13.051Z"
	}
	},...

	]
Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

Create Pet

POST https://baas.kinvey.com/appdata/***app_id***/pets	
Request headers	Authorization: Kinvey authtoken Content-Type: application/json

Request body	{
	"name": "Pepi",
	"description":"This is my 2 years old parrot named Pepi. He sings all the time.",
	"imageURL":"http://pngimg.com/uploads/parrot/parrot_PNG723.png",
	"category": "Parrot",
	"likes": "0"
	}

Response
201 Created
	{
	"_id": "59931398996ab5127d2a84d1",
	"name": "Pepi",
	"description":"This is my 2 years old parrot named Pepi. He sings all the time.",
	"imageURL":"http://pngimg.com/uploads/parrot/parrot_PNG723.png",
	"category": "Parrot",
	"likes": "0"
	}

Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

Edit Pet

PUT https://baas.kinvey.com/appdata/***app_id***/pets/***pet_id***	
Request headers	Authorization: Kinvey authtoken Content-Type: application/json
Request body	{ "name": "Pepi", "description":"This is my 2 years old parrot named Pepi. He sings all the time.",

	"imageURL":"http://pngimg.com/uploads/parrot/parrot_PNG723.png", "category": "Parrot", "likes": "0" }

Response
200 Ok
	{
	"_id": "59931398996ab5127d2a84d1",
	"name": "Pepi",
	"description":"This is my 2 years old parrot named Pepi. He sings all the time.",
	"imageURL":"http://pngimg.com/uploads/parrot/parrot_PNG723.png",
	"category": "Parrot",
	"likes": "0"
	}

Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

Delete Pet

DELETE https://baas.kinvey.com/appdata/***app_id***/pets/***pet_id***	
Request headers	Authorization: Kinvey authtoken

Response
200 OK
	{
	"count": 1
	}

Error response 404 Not Found	{ "error": "EntityNotFound", "description": "This entity not found in the collection", "debug": "" }
Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

My Pets

GET https://baas.kinvey.com/appdata/***app_id***/pets?query={"_acl.creator":"${***user_id***}"}	
Request headers	Authorization: Kinvey authtoken
Response 200 OK	[{ "_id": "5bb08d1353e0a859203643e6", "name": "Timmy",

	"description": "This is my cat Timmy. He is 2 months old and is very playful.йкнкй",
	"imageURL": "https://d3pz1jifuab5zg.cloudfront.net/2015/09/04194922/....jpg",
	"category": "Cat",
	"likes": "2",
	"_acl": {
	"creator": "5bb0874caccb650325e0227b"
	},
	"_kmd": {
	"lmt": "2018-10-01T10:35:13.712Z",
	"ect": "2018-09-30T08:45:07.709Z"
	}
	},
	{
	"_id": "5bb0888ee1d636567dc9f627",
	"name": "Yara",
	"description": "My dog Yara. 2 months old. doggy hjkhjk",
	"imageURL": "https://cdn1-www.dogtime.com/assets/uploads/2017/....jpg",
	"category": "Dog",
	"likes": "3",
	"_acl": {
	"creator": "5bb0874caccb650325e0227b"
	},
	"_kmd": {
	"lmt": "2018-10-03T08:04:40.812Z",
	"ect": "2018-09-30T08:25:50.963Z"
	}
	}]

Error response 401 Unauthorized	{ "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" }

3. PetMyPet – HTML and CSS

You are given the Web design of the application as **HTML** + **CSS** files.

- Initially all views and forms are shown by the HTML. Your application may **hide** by CSS (display: none) or **delete** from the DOM all unneeded elements or just display the views it needs to display.
- You may render the views / forms / components with **jQuery** or **Handlebars**.

Important: don't change the elements' **class name** and **id**. Don't rename form fields / link names / ids. You are **allowed** to add **data attributes** to any elements. You may modify **href attributes** of links and add **action/method attributes** to forms, to allow the use of a routing library.

4. PetMyPet Client-Side Web Application

Design and **implement** a client-side front-end app (SPA) for managing **pets**. Implement the functionality described below.

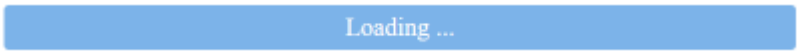
Notifications (5 pts)

The application should notify the users about the result of their actions.

- In case of successful action an **informational (green) notification message** should be shown, which disappears automatically after 3 seconds or manually when the user clicks it.

- In case of **error**, an **error notification message** (red) should be shown which disappears on user click.

- During the AJAX calls a **loading notification message (blue)** should be shown. It should disappear automatically as soon as the AJAX call is completed.



- **NOTE:** you get all the point if **all** of the notifications have the **exact content** as described in each section below

NavBar (5 pts)

Implement a **NavBar** for the app: navigation links should correctly change the current screen (view).

- Clicking on the links in the **NavBar** should display the view behind the link (views are sections in the HTML code).
- Your application may **hide** by CSS (display: none) or **delete** from the DOM all unneeded elements or just display the views it needs to display.
- The given „**Dashboard, My Pets, Add Pet, Welcome and logout**" should be visible **only** for logged in users.

- Anonymous users can **only** view the **login and register**.

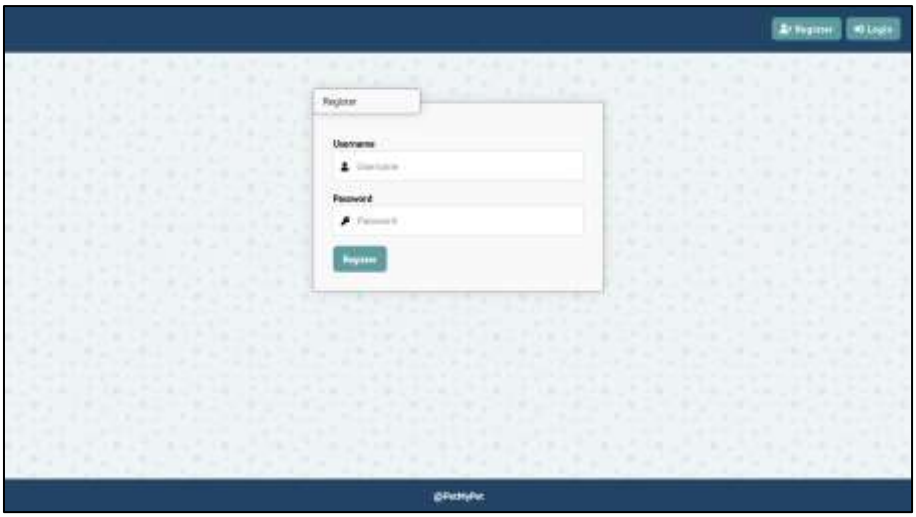
Home Screen (5 pts)

The initial screen should display the register, login and the initial image + footer.

Register User (10 pts)

By given **username** and **password** the app should register a new user in the system.

- The fields should be **non-empty**. The **username** must be **at least 3 symbols**, and the **password at least 6**.
- After a **successful registration**, a notification message **`"User registration successful."`** should be displayed and the **home page should be displayed again but with the right navbar**.
- In case of **error** (eg. invalid username/password), an appropriate error **message (`"Username must be at least 3 symbols"`, `"Password must be at least 6 symbols"`)** should be displayed and the user should be able to **try** to register again.
- Keep the user session data in the browser's **session storage**.

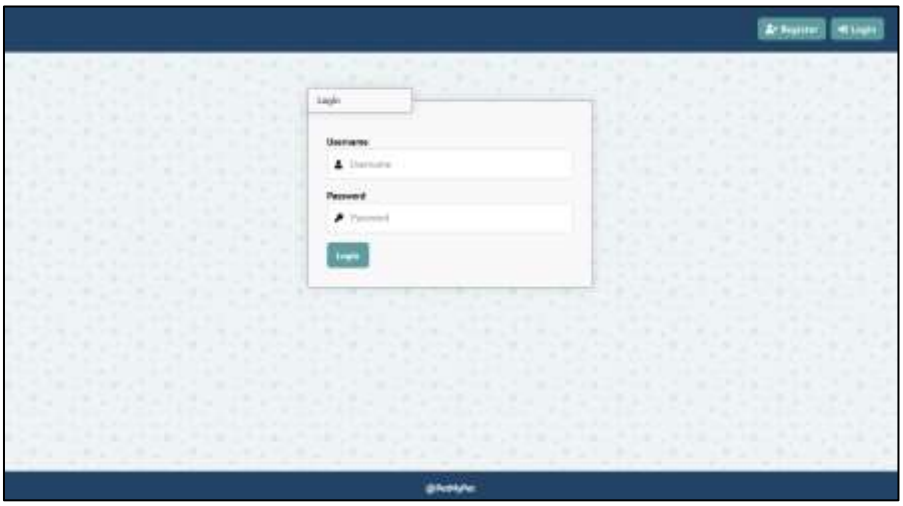


Login User (5 pts)

By given **username** and **password** the app should be able to login an existing user.

- The validations of the fields must be as in **"Register"**
- After a **successful login**, a notification message **`"Login successful."`** should be displayed and the user home screen should be displayed.
- In case of **error**, an appropriate error message should be displayed and the user should be able to fill the login form again.
- Keep the user session data in the browser's **session storage**.

- Clear **all** input fields after **successful** login.



Logout (5 pts)

Successfully logged in user should be able to **logout** from the app.

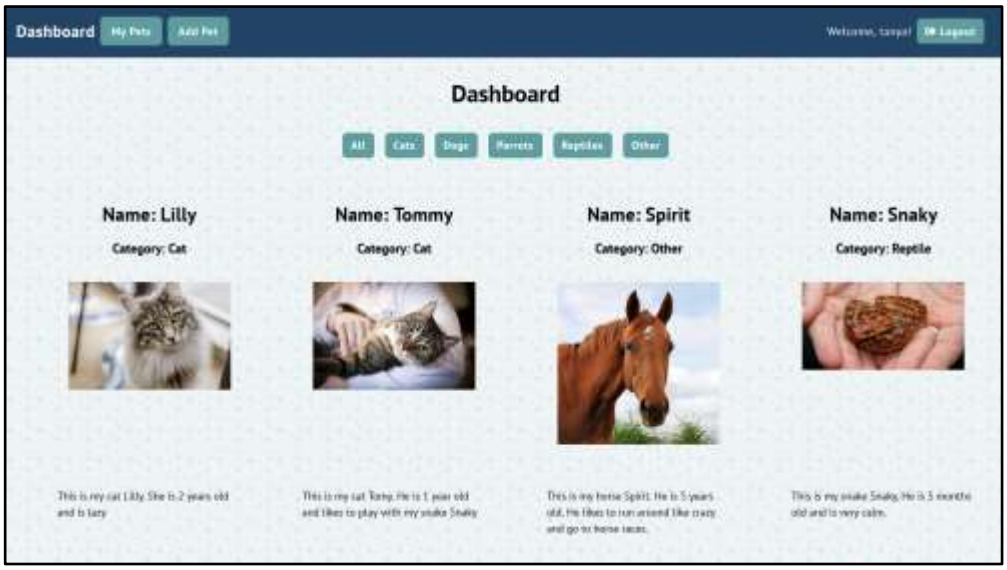
- After a **successful** logout, a **notification** message **`"Logout successful."`** should be displayed.
- After successful logout, the **anonymous screeen** should be shown.
- The **"logout" REST service** at the back-end should be obligatory called at logout.
- All local information in the browser (**user session data**) about the current user should be deleted.

Logged In Dashboard (30 pts)

1. All Pets (15 pts)

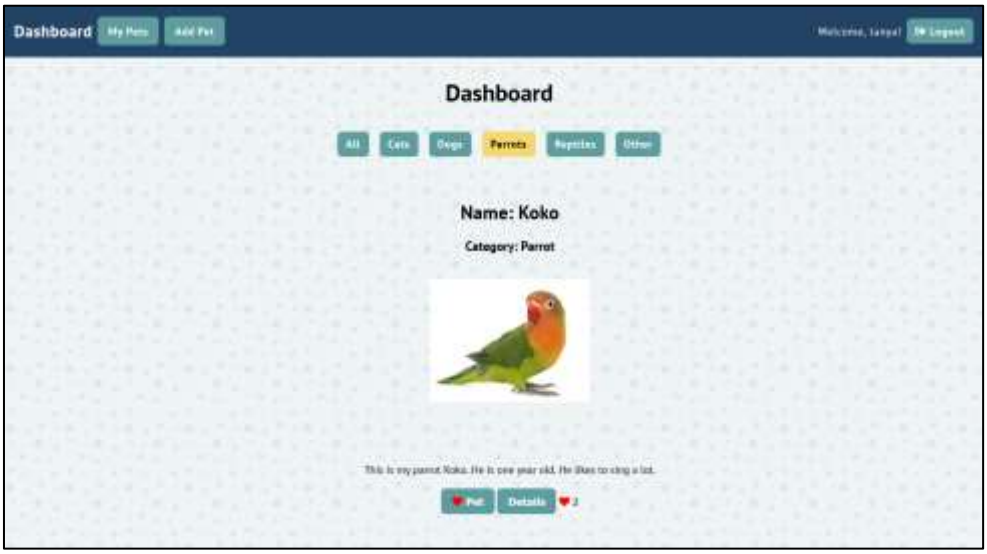
Successfully logged users should be welcomed by the **home screen**. They should also be able to see the dashboard, which must have the following:

- The **pets** should be listed in the **format** as shown in the Web design (see the screenshot below).
- Only the pets that are **not created by the current user** should be listed!
- Each **pet** has **name**, **image, description**, **category**.
- When the **details** link is clicked, your app should **redirect** to the details **section** of the pet.
- When the **pet** link is clicked, the **"pet counter" of the pet should be increased (it should start from 0)**



2. Categories (15 pts)

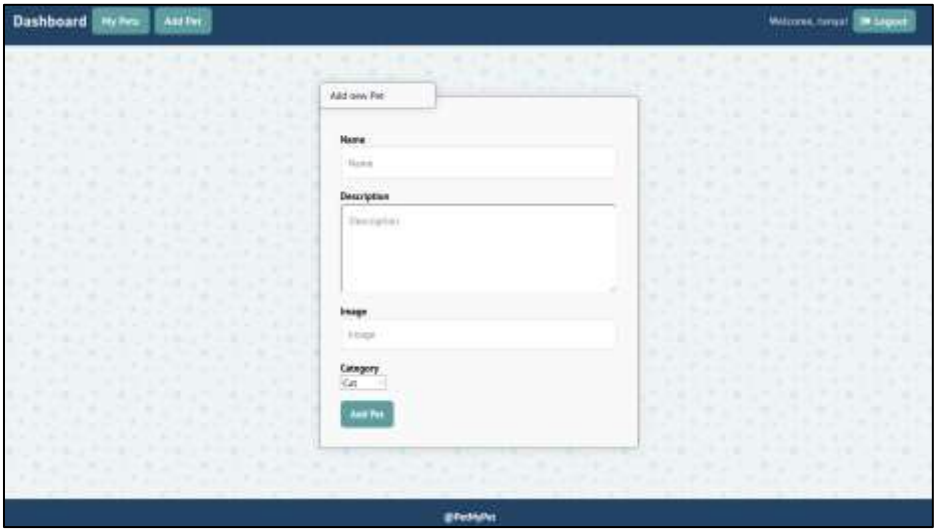
- When clicking on some of the **categories**, the pets should be **filtered by the selected category** and **oredered by their "pet counter" in descending**



Add Pet Screen (10 pts)

Logged in users can **add** pets. Clicking the **[Add Pet]** link should open a form.

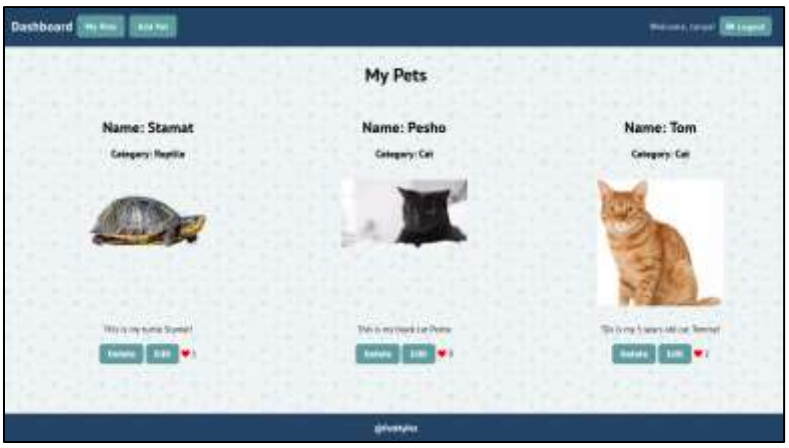
- After a **successful** pet creation, a notification message **`"Pet created."`** should be displayed and the **home page** should be shown.



My Pets (10 pts)

Each **user** should be able to view his own pets by clicking [**My Pets**].

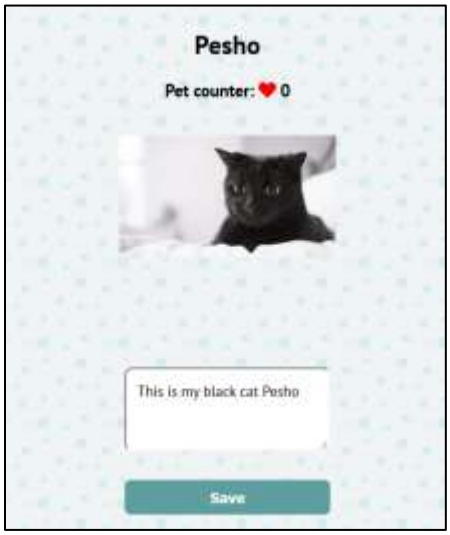
- The **pets** should be listed like in the **dashboard** section, except the **"Pet"** button is replaced with **"Delete"** button (users cannot pet their own pets) and the **"Details"** button **is now "Edit"** button. **No ordering needed**.



Edit/Details Pet Screen (5 pts)

Users **should** be able to **edit** their own **pet's description**. Clicking the [**Edit**] link on each pet should open a details page. Inside there should **not be a "Pet" button** and the **description should be displayed in a text box**. There should also be a **"Save"** button to save the changes.

- After a **successful** pet update, a notification message **`"Updated successfully!"`** should be displayed and the **dashboard** should be shown.



- Users should be able to see **[Details]** button on **other pets** and they should be able to **"Pet"** them, **but not edit** them:

Pet a pet (5 pts)

Every user should be able to **pet other pets**, but **not his own**. By clicking on the `[Pet]` button, **the counter of each pet increases**.

Delete Pet (5 pts)

Owners **should** be able to **delete** their **own** pets by clicking the [**Delete**] button.

- When the button is clicked, display the following page:

- When the button **"Delete"** is clicked, the pet should be deleted
- After **successful** pet delete a notification message **`"Pet removed successfully!"`** should be displayed and the **home catalog** should be **shown**.

5. Subtmitting Your Solution

Place in a **ZIP** file your project folder. Exclude the **`node_modules`** folder. Upload the archive to Judge.