Adopt-A-Pet Technical Report

PREPARED BY

Sean-Matthew Aujong

Tushar Kohli

Nandu Vudumula

Edward Chamblee

Sohum Kapoor

Motivation

Adopt-A-Pet serves as a one-stop website where potential pet owners research the best way to adopt their next pet. From lifestyle preferences to browsing adoption centers, this product speeds up the timeline from visiting the adoption center to bringing the pet home. Adopt-A-Pet emphasizes enabling compassionate pet owners and underserved adoption candidates. In particular, Adopt-A-Pet will provide information for adoption centers, different species and breeds, as well as adoptable pets.

User Stories

Phase I Stories

Choosing Pet by Breed

“I am a pet lover who is looking for a specific dog breed to adopt. I would like to see all the dog breeds available for adoption and select one or more of the available dog breeds. I only want to see dogs of the breeds that I selected.”

Status: Phase 3 - Adoptable Pets now has a Breed filter.

Distance to Adoption Center

“I would like to adopt a pet in the nearest adoption center. I would like to be able to find adoption centers near me (by distance or P.O. box). I would also like to be able to sort this list by adoption centers nearest me.”

Status: Phase 4 - Narrow down adoption centers by zip code

Filtering/Sorting Pets by Adoption Center

“I would love to adopt a pet, however I only want to adopt from no-kill shelters because I believe they are more humane. I would love to be able to filter or sort the adoptable animals by what type of animal shelter they come from, or by the rating of the animal shelters they come from.”

Status: Phase 3 - Adoption Center page now has a Type filter, which includes No-Kill.

Have Images of Pets on Their Page

“I don't know much about different breeds of dogs, so I won't be searching for a specific breed. My kids only want to adopt a dog if it looks a certain way. It would be really helpful to see images of what the available pets look like.”

Status: Phase 1 - The instance page for an Adoptable Pet now contains its picture.

Looking for a Specific Animal

“I am looking for only cats, and want to be able to filter the results so no other animals appear in order to make my experience faster.”

Status: Phase 3 - Species/Breeds model is now filterable between cats and dogs.

Phase II Stories

Filter by Breeds

“I am really interested in adopting a pet for my household. More specifically, I am looking to adopt a beagle. I would like to see a general view of all beagles that are currently up for adoption.”

Status: Phase 3 - Search can be used for the “beagle” keyword.

Filter by Adoption Center Location

“I am thinking of adopting a pet from an adoption center in my area. It would be great to see a list of adoption centers filtered by PO Box. If this is not possible, I'd love to see it by county or city.”

Status: Phase 3 - Adoption Centers are now sortable/filterable by state, zip code, and city

Adoption Center Map

“I am considering driving to a nearby adoption center. It would be great to have a map associated with each adoption center so that I could get an idea of the area it is in.”

Status: Phase 2 - Each adoption center instance now includes Google Maps media.

Multiple Images of Species

“I am not that knowledgeable when it comes to dog species and would love to see images to gain a better understanding. Would it be possible to have multiple images of each dog species? A variety of sizes and colors would be great.”

Status: Phase 4 - Generated more YouTube media types for species

Medical Information

“I noticed that a feature on the pets page was medical information. I was wondering if it would be possible to see that earlier, without having to click on each pet to find out.”

Status: Phase 4 - Medical info is sensitive, implemented general energy and intelligence levels instead

Phase III Stories

Refined Images

Website is looking good guys. However many of the images on the site look a bit distorted. The pets would look a lot more appealing if the images were resized a bit so they did not look stretched out.

Status: Phase 3 - fixed About page image formatting, fixed formatting of pet images.

Remembering Pagination upon Refresh

Hello, I am really interested in adopting a pet. I love the layout of the website, but I am running into an issue when browsing the available pets. The page number that I was browsing is not remembered if I go back to that page from a pet's page. For example, let's say I was on page 10 and click on a specific pet. If I press the back button from their page, it brings me back to page 1 of the "Available Pets" page instead of page 10. It would be nice if it brought me back to the page I was on so I can resume looking at pets on that page. Thank you so much!

Status: Phase 4 - Pagination is now remembered in user session

Page Navigation Bar Convenience

As I was scrolling through the list of adoptable pets, I noticed that the page navigation bar was fixed at the top. This made it a little uncomfortable to have to scroll all the way back up to go to the next page. I think it would be better if there were page navigation bars at the top and bottom.

Status: Phase 4 - Decreased page length to compensate for nav bar

Sparse Instances

As I was looking for the different species, I noticed that some instances would have very little data on it. For example, https://www.adoptapet.me/sbmodel/16, this instance has just one field filled in. Perhaps it would be better if such instances were simply omitted.

Status: Phase 4 - Decided that sparse pages are actually necessary for now, added more data

Impossible Dropdown Menu

When filtering on adoption centers, I noticed that the 'city' downdown menu comes before the 'state'. This can lead to some unreachable filters, such as city == Austin and state == NY. Obviously such location doesn't exist and so this can be misleading for users. It would be great if city and state (and even zip code for that matter) were dependent on each other for accurate filtering.

Status: Phase 4 - Just using the City dropdown menu for our dataset should be sufficient

User Stories for Developer Group (Gallery Gaze)

Phase I Stories

Awarded Artists

I am a casual art enthusiast who wants to learn more about artists who have received a certain award. I would like to see all the different awards possible and filter artists by awards received. It would be really cool to see what kind of artwork they won the award with, as well as a text and image description of the award.

Researching Artistic Movements

I am a student researching a particular art movement. I should be able to search up artworks and artists from a certain art movement. I want to know about the time period that the art movement was in, as well as its common characteristics and historical motivations.

Visiting Museums

I am a vacationing tourist looking for museums in the area I am vacationing in. I would like to search for museums in a certain radius, as well as their operating hours. I want a description of the museum as well as a general overview as to what artists and artworks it hosts.

Art Origins

I am a college student in an art history class and I would like to learn about art pieces and artists that originated in a certain area. I want to filter artworks and artists by some geographical location (country, city, etc.). I would like to know about a description of their origins as well as dates.

Fame in Art

I am a social media influencer looking to make content on the top echelon of wealth and fame in the art world. I would like to sort prominent artwork, artists, and museums by some measure of net worth. I would also like to see a heatmap of prominently successful origin locations.

Phase II Stories

Art Media

I am a hobbyist learning to use a particular art medium. I would like to be able to filter artworks by medium. I would also like to filter artists by which medium they are famous for using.

Gallery Sizes

I am a traveller planning a trip to see galleries of different sizes. I would like to be able to search for galleries based on the number of works they host and how many people visit daily.

Cost of Museum

I am a traveling tourist looking for museums to visit. I want to pick the museum I want to go to based on how expensive the museum is. I would like to filter and sort the list of museums by price.

Art Materials

I am an art student trying to learn about artworks made using certain materials. I would really like to know why certain materials are used. I would like to also search for artworks by the materials they are made with.

Museum Popularity

I am an avid art enthusiast touring famous museums. I would like to sort the museum list by yearly visitors. I would also like to sort a list of museums by yearly visitors around a certain area.

Art Timeline

I'm an art teacher preparing a timeline of different artworks. I would like to be able to view a timeline of artworks. Based on their date of creation, I would like to generate this timeline.

Phase III Stories

Address Visualization

I am a casual tourist looking for museums to visit. I am far too lazy to actually visit the instance pages themselves. Given an address on the model page, I would like to be linked to a Google Maps entry on a new tab.

Phone Number

I am a user looking to contact museums for more info. I see that the model page has a list of phone numbers. On my phone, I would like to click on that phone number to open up my phone app to call that number.

Website Links

I am a customer looking to find the website of the museum. While Gallery Gaze currently offers these links, there is no way to click on them from the model page. I would also like a way to click on the links from the instance page as well.

Google Maps Marker

I am a user trying to find out where a museum is on Google Maps. While the instance page provides a Google Maps instance, it lacks a red marker. A red marker would help me figure out where exactly the museum is.

Museum Status

I am a user trying to figure out if a museum is operational. When filtering for museums by status, I am unsure by what my options are. I wish there was some diagram on the model page where I could see what the different status options are.

RESTful API

https://documenter.getpostman.com/view/17710041/UUy38kte

Adoptable Pets API Endpoints

GET all adoptable pets

api.adoptapet.me/ap

Returns a list of all the adoptable pets in our database with basic information on the adoptable pet, such as its name, sex, age, color, description and a url to a picture of it (and the adoption center number it belongs to as well as the breed number it is).

GET adoptable pet by api_id

api.adoptapet.me/ap/<int:api_id>

Returns the information of an adoptable pet given its (api_ (meaning from api.rescuegroups.org))id, such as its name, sex, age, color, description and a url to a picture of it.

Breeds API Endpoints

GET all breeds

api.adoptapet.me/sb

Returns a list of all of the breeds in our database with information on each breed, such as its name, affection level, alternative names, api_id, adoption centers where it can be found, child-friendly, code of country of origin, dog-friendly, energy level, grooming difficulty, hairless, health issues, whether it is hypoallergenic, ID, intelligence level, life span in years, country of origin, shedding level, short legs, social needs, species ID, species name, stranger-friendly, suppressed tail, temperament, vocalization, weight, Wikipedia URL, and youth name.

GET breeds by api_id

api.adoptapet.me/sb/<int:api_id>

Returns the information of a breed given its api_id, such as what centers carry that breed, the adoptable pets that are of this type of breed, the name of its species, the name of the breed, what a youth of its species is called, its temperament, life span, alternate names, wikipedia url, origin countries, weight, country code of its country of origin, and height. There are also binary values for traits such as hairless, suppressed tail, short legs and hypoallergenic. On the other hand, the following characteristics are on a scale from 1 to 5, with 5 being the maximum or highest: adaptability, affection level, child friendly, dog friendly, energy level, grooming, health issues, intelligence, shedding level, social needs, stranger friendly and vocalization.

Adoption Center API Endpoints

GET all adoption centers

api.adoptapet.me/ac

Returns a list of all the adoption centers in our database. This information includes what pets it shelters, the species of those pets, the center's name, street address, city, state, zipcode, email, phone, latitude and longitude coordinates, and the services it provides

GET adoption center by api_id

api.adoptapet.me/ac/<int:api_id>

Returns the information of an adoption center given its api_id. This information includes what pets it shelters, the species of those pets, the center's name, street address, city, state, zip code, email, phone number, latitude and longitude coordinates, and the services it provides.

Models

Breeds

- api_id
- centers
- pets
- species_id
- species_name
- breed_name
- youth_name
- temperament
- life_span
- alt_names
- wikipedia_url
- origin
- weight
- country_code
- height
- hairless
- natural
- suppressed_tail
- short_legs
- hypoallergenic
- adaptability
- affection_level
- child_friendly
- dog_friendly
- energy_level
- grooming
- health_issues
- intelligence
- shedding_level
- social_needs
- stranger_friendly
- vocalization

Adoptable Pets

- api_id
- center_id
- species_breed_id
- center_number
- breed_number
- name
- sex
- age
- color
- desc
- pic_url

Adoption Centers

- api_id
- pets
- species_breeds
- name
- street
- city
- state
- zipcode
- email
- phone
- latitude
- longitude
- services

Filterable/Sortable Attributes:

Breeds: name, breeds, life expectancy, weight (size group: small, medium, large), origin

Adoptable Pets: species, breed, age (group), sex, size (group)

Adoption Centers: state, country, species carried, service(s) offered, type (rescue v shelter)

Searchable Attributes:

Breeds: name, country of origin, color, size group, life-expectancy

Adoptable Pets: species, breed, age (group), sex, size (group)

Adoption Centers: state, country, species, service provided, type

Media:

Breeds: google map of origin country, youtube videos, table of breeds, text list of adoptable pets of that breed and adoption centers that carry that breed

Adoptable Pets: grid of pets, picture of pet, text of its characteristics and a description of it, map of its Adoption Center location

Adoption Centers: table, map of where its located, text of its services and type, text list of the species it carries, and tags of the services it offers

Connections:

Species: Connects to Adoptable Pet because there will be multiple Adoptable Pets of a certain species. Connects to Adoption Center because an Adoption Center carries multiple species.

Adoptable Pets: Connects to Species because an Adoptable Pet is of a certain type of species. Connects to Adoption Center because each Adoptable Pet is found under its respective Adoption Center.

Adoption Centers: Connects to Species because an Adoption Center will carry certain types of Species. Connects to Adoptable Pets because an Adoption Center will have certain Adoptable Pets available.

Tools

Backend

- AWS Relational Database Service (RDS): Used to host our database in the cloud.
- Elastic Beanstalk: Used to host and deploy backend API.
- Postman: Used to test API calls and create documentation of our API.
- NameCheap: Used to get a domain and for DNS management.
- SQLAlchemy: Used to interact with the database through the filter() and order_by() functions.

Frontend

- React: Used to render the website and connect the user interface to the backend.
- React-Highlight-Words: Used to search for substrings of text.
- React-Select: Used to handle the sorting/filtering dropdowns on the model pages.
- React Bootstrap: The CSS framework used for the website.
- Selenium: Used to test the GUI of the website.
- Enzyme: Used with Jest to simulate onClick events
- Jest: Used to test React and Typescript components and functions.
- AWS Amplify: Used to deploy the frontend of the site.
- GitLab: Used for code storage and CI/CD.
- D3 - Javascript library for building visualizations like bubble charts
- Recharts - Based on typescript and is another library for building visualizations like pie charts

APIs:

- GitLab API - Serves statistics for About page, such as commits and issues by user.
- RescueGroups v5 - Serves Adoptable Pets and Adoption Centers and Species and Breeds. It provides at least five attributes for each of those models, except species and breeds.
- The Cat API - Serves generic Cat information for Species/Breeds. Serves to supplement the attributes we have for all cat breeds to give us many more than just five.
- The Dog API - Serves generic Dog information for Species/Breeds. Serves to supplement the attributes we have for all dog breeds to give us many more than just five.
- Google Maps API - Serves Google Maps embeds for various instance pages-- for example, the adoption center instance pages. The user must click on a button for the map to correctly drop a pin at the adoption center location.

Hosting

We acquired our domain name from NameCheap for free using the GitHub Student Developer Pack. We deployed our GitLab repository on AWS Amplify, which hosts our web content. AWS Amplify connected to our domain name from NameCheap and also issued the TLS/SSL certificate to enable HTTPS protocol. We also handled redirection from *adoptapet.me* to *www.adoptapet.me*.

Phase 2 Features

Pagination:

We implemented pagination in the backend by referencing the code of a past group, Texas Votes. We first enabled the client or user to pass query parameters to our three API endpoints, /ap, /ac and /sb. We look for ‘page’ and ‘perPage’ query parameters specifically. Then we query the table for each respective endpoint. If the user doesn’t specify a page manually, then we default to returning the first page of data. We store the count of entries we have in our db for the db query, and then if the user specifies the perPage param, we store that otherwise we return a default value of 20 instances on each page. Finally, we return the instances and count of total number of instances.

Database:

We use the AWS Relational Database Service (RDS) and connect it to an instance of a PostgreSQL database so we can store our data in the AWS cloud. We used pgAdmin as it provides a GUI with which we can see and query our database and tables therein. We specified our AWS RDS endpoint as the pgAdmin host and connected to our database with the username and password we created. Flask knows how to connect to our database because we specify it in the app configuration. We used Flask-SQLAlchemy in order to populate, query and filter our database and Flask-Marshmallow to provide a schema for all the tables inside our database. These schemas are what specify what fields in the JSON object our API endpoints return.

Phase 3 Features

Filtering:

Our backend API implements filtering functionality. We use SQLAlchemy's filter() function on the BaseQuery object we get from querying a particular table. We define a set of criteria that the user can filter by and filter one attribute criterion at a time. We also allow the user to filter by more than one value for a single attribute (think /adoptablepets?color=black&color=white), and this has the behavior of including items that meet at least one value of attribute.
On the other hand, if the user chooses to filter by multiple different attributes (color=black&size=medium), then the filter will only admit objects that meet all of the filter criteria.

The front end just calls the back endpoints depending on what the user chooses in the filters, and displays that data. We used the Select component from the 'react-select' package to provide all the filtering options. The user can mix and match multiple attributes and delete any or all of the options, and the data will be filtered and returned appropriately.

Sorting:

Similarly, the backend API also implements the sorting functionality. We take a column to sort by (age or color, e.g.) and use the order_by method in SQLAlchemy. This function reorders the rows according to the attribute and only one attribute is allowed for a sort. We used the interesting case expression in SQLAlchemy when we wanted the particular levels or values of an attribute (say, small, medium and large for pet size) to be in a particular order (small first, then medium and large)

Again, the front end just calls the backend API endpoints for the sorting options we are using. The sorts are implemented the same as filters using the Select component from 'react-select'. On the pets model page we have 5 unique filters and a sort that lets you sort 4 of those filter options. On the centers and breeds model pages we have 4 unique filters and and one unique sort for each page.

Searching:

Searching too we implemented in the backend API. Searching is a bit more advanced but it is essentially filtering on possibly many search terms. We split the user provided terms in the q= query parameter using the Python split() method which splits by whitespace. Then for each term we append the underlying sql expression for matching up to the five attributes we permit into a list. This list then we initialize a tuple with, then unpack it providing it to the or_() function is SQLAlchemy which itself inside a filter function. This has the effect of filtering by all of these terms. An interesting addition was to pass in another tuple initialization then unpacking for a list of the breed names that match any term the user provided. We utilized Python's feature of list comprehension for this, which was pretty cool (with help from the TA's). Importantly a join had to be done on the table that had the specific attribute we wanted to filter by (breed_name in BreedSpecies when we're searching for Pets)

Yet again, the front end just calls the backend API endpoint to get the results for a search query. On each model page there is a search form that the user can use to get results for that specific model. There is a general search page that uses the same search form, and then concatenates the results from all model pages through the backend API endpoint.

Phase 4 Features

Visualizations:

Visualizations were generated through d3.js and recharts. Our bubble chart specifically uses d3.js, but the rest of our visualizations such as the pie chart and funnel chart use recharts instead. Data was pulled from both our adoptapet.me API and our developer group's gallerygaze.me API. We leveraged a previous optimization we made for our own visualizations where we used our search functionality to only return certain columns, making the load times much faster.

Gitlab

https://gitlab.com/10am-group-8/adopt-a-pet