Milestone 5
Due Dec 14, 2021

Project Milestone 5 - Animal Shelter Index

The (sorta) CS Majors
Kate Sargent | A02232867
Jonas Williams-Gilchrist | A02268191

Application URL
https://is-5050-shelter.herokuapp.com/

Application Summary

Our application is an animal shelter index. The purpose of our application is to provide users with an easy way to view pets they may be interested in adopting. Additionally, our website provides various other services centered around the adoption of pets, such as discussion boards, news, events, information about the animal shelter, and a link to donate to the local shelter.

The intended users of our application are people who want to adopt an animal, want to put an animal up for adoption, and want to learn more about their local shelter and community involvement. These users have several ways to dynamically interact with our website and its pages.

The application has several different features related to the pet adoption process. First and foremost, users can search through our database of current animals at our shelter and adopt an animal through that animal's specific page. Additionally, a user can list their own pet for adoption by creating a form with information about their pet and uploading a photo. Furthermore, users can donate to the shelter and create discussion posts about topics related to animals and adoption.

Routing Guide

GET /index : Shows the home page of our website
GET /contact-thanks : Displayed after submitting contact info, thanking you for doing so
GET /about : Shows information about our animal shelter
GET /about/contact-us : A page to enter your email and name to be put on a mailing list
POST /contact-submit : The post location for the contact information form
GET /donate : Displays the shelter donation page
GET /about/volunteer : Provides information about volunteering for the animal shelter

GET /pets : Shows the list of all pets in our database
GET /pets/index : Shows the list of all pets in our database
GET /pets/api : Shows all pets in the database in json format
POST /pets/filter : Applies the filter options selected and re-routes to index
GET /pets/add-pet : Contains a form for adding a new pet to our database
POST /pets/postPet : Handles the form data for creating a new pet entry in the database

POST /pets/adopt/:petId/:userId : Handles adopting a specific pet by a specific user
GET /pets/adopted : Displays a message thanking you for adopting a pet
GET /pets/:id : Displays specific information about a specific pet

POST /users/create : Creates a new user account in the database
GET /users/login : Shows the user login page
GET /users/signup : Shows the user signup page
POST /users/login/authenticate : Authenticates the user trying to login
GET /users/logout : Logs the user out of their account
POST /users/submit-donation/:id : Handles a user making a donation, api call to Stripe
GET /users/account : Shows information about the logged in user, including pets, email, etc.

GET /events : Shows all events with relevant information
GET /events/add-event : Displays form for creating a new event
POST /events/create : Handles the creation of a new event

GET /news : Shows all news posts with relevant information
GET /news/add-news : Displays form for creating a new news post
POST /news/create : Handles the creation of a new news post

GET /discussions : Shows all discussion posts
GET /discussions/new-discussion : Form to create a new post
POST /discussions/create : Handles creating a discussion post
GET /discussions/:id : Shows specific information for a discussion post
POST /discussions/:id/postComment : Handles posting a comment on a discussion post

Error handling routes are also implemented if there is no valid matching route.

User Credentials and Roles

Our website has an admin designation on user accounts. Admins are able to add pets (put them up for adoption), create news posts, and add new events. Standard users are not able to access these functions.

Admin user account:
Login: kate@sargent.com
Pass: password123

Standard user account:
Login: jonas@w-g.com
Pass: 123pass

Bug Hunt Responses

Did not see anything that would be part of an API, but might not be readily visible	Implemented Stripe for donation payments
Discussion board has hard coded dates on the comment display	Added dates to the view template
Donations allowed me to donate $0 dollars, You can also add very very small amounts as in .0000000001 and such. This should likely have a minimum amount as well as the allowed number of decimals to cents.	Switched donation method to use stripe and reduce decimal places
When completely logged out and attempt to visit a URL to like /users/accounts it throws an internal server error :)	Added new controller functions to validate user logins

Project Requirements

1. 5 interlinked dynamic pages (views) that display dynamic, database-driven content per team member. To be counted as dynamic, the page must retrieve information to and/or write information to the application database using one or more queries/data access functions written by the team.	Views that display dynamic content from the database and associated queries/transactions: 1. /pets: Displays the pets 2. /pets/add-pet: Form that creates a new pet in our database 3. /users/account: Shows user information 4. /users/signup: Shows the user creation form that creates a new user in the database 5. /events: Displays events 6. /news: Displays news 7. /discussions: Displays discussions 8. /discussions/create: Creates a new discussion board in our database 9. /discussions/:id: Shows information about a specific discussion, including database information in the form of comments 10. /donate: Amount donated is recorded in the database associated with that user

2. At least 5 database transactions (a transaction includes selecting, inserting, updating, or deleting data from the database) per team member	We have many transactions that take place in our application, these are handled by POST routes and typically include adding something to our database, like creating a new user, new pet, removing a pet from the database, etc. 1. /pets/filter: Applies filters to pet information when retrieving it from the database 2. /pets/adopt/:petId/:userId: Transfers "ownership" of the pet from the pool to a specific user 3. /users/login/authenticate: Checks information against what is stored in the database to authenticate the user 4. /users/submit-donation/:id: Alters the user in the database, changing the amount they have donated. 5. /events/create: Creates a new database object for event using information submitted through a form 6. /discussions/create: Creates a new database object for create using information from a form 7. /discussions/:id/postComment: Creates a new comment on a specific post 8. /news/create: Create a news object in the database using form information 9. /pets/postPet: Creates a new pet object in the database using form data 10. /contact-submit: submits contact information
3. Support for user authentication and secure storage of user credentials	User credentials are managed through the Passport.js package. Users authenticate through a local authentication strategy that verifies credentials through a mongoose model
4. Preservation of state (may include cookies, session state, url parameters, querystring, etc.)	We use cookies and store a session to keep track of useful information about the user that our application needs.
5. Development of at least one REST API and consumption of another third-party API	We utilize the Stripe api to process payments through. Also, we expose all of the pets currently in the database with the route

	/pets/api.
6. Inclusion and use of at least two npm libraries that are not covered elsewhere in the course (i.e., you need to explore additional functionality on your own).	We used some of the morgan package to record information about data requests, but decided to remove it along the line as it became too verbose for what we needed. We also used bs4-toast
7. Input validation and error handling for all free-form user entries that could potentially result in errors error routes that handle unexpected http requests.	We have used mongoose models to validate data types for creating new database objects. Our application also