CS-499 Module 3 Milestone

Alexander Dabrow

Southern New Hampshire University

26 May 2025

The goal of this milestone was to perform the planned enhancements that aligned with the category of Software Engineering and Design.

The original artifact I chose to enhance was a console application for a rescue animal management system from an IT-145 project. The main features of the program include displaying dogs and monkeys in the system, as well as being able to make reservations for the animals to use as service animals. The goal of this enhancement is to develop this console application into a full-stack web application.

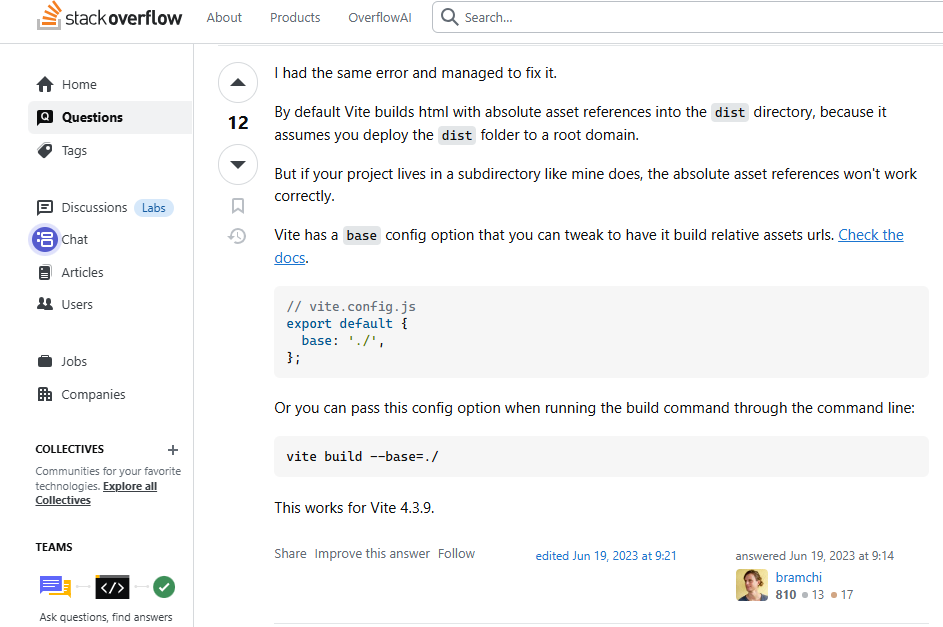
For any software engineer – I think it’s a good idea to have basic fluency in full-stack development which is the purpose why I chose this route. You have data you need to manage and manipulated but you need to be able to display that in simple and easy to understand UX

For the front end, I used React with Vite along with TailwindCSS. I then configured continuous deployment to Azure Static Web Apps using GitHub.

I have made a personal project before using React, so I came in with some foundational knowledge with props, React Hooks such as useState and useEffect, and using features like ternary operators for conditional rendering within components.

Local development while building it was straightforward. This was my first time using React with TypeScript, so some additional care was needed when defining and using types.

Everything was fine until I went to deploy to Azure Static Web Apps. The last time I made a React project, I used JavaScript. I didn’t run into the issues I did when I deployed this time. It was tough to figure out what was causing the issue – nothing was loading on the screen and the only console error I was getting was:  
Uploaded image

The answer I found from searching on Stack Overflow -

<https://stackoverflow.com/questions/75433591/failed-to-load-module-script-expected-a-javascript-module-script-but-the-server>

The last time I used React, I did not use Vite so this was a novel issue for me.

A screenshot of a computer program

AI-generated content may be incorrect.  
I was getting errors from unused variables and parameters , so I did reference ChatGPT for the following changes to the tsconfig file here - there are temporary changes and only for testing purposes - noUnusedLocals and noUnusedParameters will be turned back off for the final version of the product.  
A screenshot of a computer program

AI-generated content may be incorrect.

The other changes –“ target” or the JavaScript version it compiles to, “module’ for the module system to be used, “moduleResolution” for handling imports, “strict” for type checking, “jsx” – “react-jsx” for the JSX transform rules.

Tailwind CSS. I have used Tailwind CSS before, but I have never quite used it to this degree. It took a *significant* amount of time to get everything to look the way I wanted to, but still I think was a good choice to use because of all the time saved just from being able to do almost all styling based on the className. I learned some new features like using “focus:” which allows you to define effects when an element is selected like the search box on the reserve page.  
  
A screenshot of a search box

AI-generated content may be incorrect.

A screenshot of a search box

AI-generated content may be incorrect.

className="px-4 py-2 w-64 rounded border border-gray-600 bg-slate-800 text-white placeholder-gray-400 focus:outline-none focus:ring-2 focus:ring-teal-400 transition duration-200 ease-in-out"  
  
So first we define the basic horizontal and vertical padding, make the box rounded, add a default border, set the border color, set the background color of the box, set the text the user types in to white, and set the placeholder text to gray. After that we define the focus behavior when a user clicks on the box. First we display a ring with a thickness of 2, set the ring color to teal, and then we define a transition, its duration and a function to transition it smoothly (ease-in-out). Doing transitions and changes with user activity I think makes the site feel more responsive and dynamic.

A screenshot of a computer screen

AI-generated content may be incorrect.

Transparency effect – My original color scheme (above) was going to be with this shade of fuchsia, but then I started to tinker with the transparency values. So, the “See all dogs” box right now is defined above as className= “bg-fuchsia-600…”, but then if that number is divided, it sets the opacity value.

A screenshot of a computer

AI-generated content may be incorrect.

This was the box set to an opacity value of 20. It didn’t feel like it was any more transparent – it felt more like it was just changing the color. I then figured it might just be nicer to transition on hover like the box is turning transparent.

Like the “focus:” element, I used “hover:” which lets you define styling when the user hovers over that element. First, we define the transition for all the properties that are going to change, set the duration of those changes, ease those changes in and out, set the values to be changed on hover, which included adding a shadow and scaling the box larger by 5%.

This was done with: “… transition-all duration-200 ease-in-out hover:bg-gray-900/15 hover:scale-105 hover:shadow-lg hover:shadow-gray-700”

Normal:

A screenshot of a computer

AI-generated content may be incorrect.

On hover:

A screenshot of a website

AI-generated content may be incorrect.

Small features like this contribute to the simple and clean looking UI the site has.

A screenshot of a social media account

AI-generated content may be incorrect.

Source: https://www.youtube.com/watch?v=5SnJ7Vyf9zs

I took inspiration for the text styling from YouTube in utilizing different font sizes, emphasis, as well as colors/shades.  
A screenshot of a phone

AI-generated content may be incorrect.

For functionality:  
  
The file structure for the frontend is setup according to the screenshot below:  
A screenshot of a computer

AI-generated content may be incorrect.

For the components folder, this is where components of a page are stored. These are reusable chunks of code designed to render a part of a page. I then would call these components either in App.tsx (for continuously display the header and footer), or in one of the pages from the pages folder.  
  
Design choice – I could have done AnimalList instead of both DogList and MonkeyList, but that would take some additional configuration setting up the conditions to make that work, but it wouldn’t be a bad choice if more animal types were going to be added and we didn’t want a list file for each animal.

The Routes folder is simply for defining the routes.  
A computer screen shot of a program code

AI-generated content may be incorrect.  
We first define the path that is appended to the URL.  
So if our base URL is <https://wonderful-sky-03a377e10.6.azurestaticapps.net/>, then <https://wonderful-sky-03a377e10.6.azurestaticapps.net/dogs> will then call the corresponding element which is DogPage.  
  
Then we have our Types folder.  
 A blue background with white dots

AI-generated content may be incorrect.   
This is where we define the data structures for our animals.  
A screen shot of a computer program

AI-generated content may be incorrect.  
A screen shot of a computer code

AI-generated content may be incorrect.  
We define attributes in the parent class of Animal and then the child classes, dog and monkey, inherit those and then define their own class specific attributes.  
  
For now, I added the question mark after the attributes to say it’s okay to have undefined values and really that is only for testing purposes.