Todd Spainhour

Final Project Report
Doggy Details
SWE 6623 – Summer 2023

GitHub Repo: https://github.com/ToddSpainhour/DoggyDetails
Frontend Port Number: 5500 Backend Port Number: 7260

Proposal

Our beloved pet, Scout, recently injured his leg, and after a recent visit to the veterinarian's office, we learned he tore a ligament that will require surgery. I've noticed my wife and I communicating a lot about Scout's details such as if he's had his medicine, eaten his food, and summaries about his short, slow-paced walks. This situation is the premise of my project proposal which is to build an app that tracks all those details and shows trends in the data. Overall, this project was good training, and I learned a lot from the experience.

Technical Details

This project will consist of three main parts: frontend, backend, and a local database.
Frontend: HTML, CSS, and JavaScript.
Backend: ASP.NET Core Web API in C#.
Database: SQL
Other Technologies: Visual Studio, Visual Studio Code, Azure Data Studio, GitHub, GitHub Desktop, Live Server, and Swagger

Testing

For this project I've used manual unit, integration, and regression testing. I encountered some common errors like click events not firing, trouble with exporting JavaScript modules, Cors issues, and an http method sending back a status code 400 for a Bad Request. I encountered a bug that I assumed was related to timing which led me down the wrong path to finding a fix.

I made a simple authentication solution where after someone creates an account or logs in, it saves their unique primary key, OwnerID, from the database into the browser's cookies. This value is used as a parameter when querying the database to return data specific to the authenticated user.

When the dashboard page loads, I run an async function called getAllPetsForThisOwner() which grabs their OwnerID value from cookies and uses it as a parameter in the query. Occasionally, the query would return no results and the user was presented with the message, ""You have no pets in the database" when results should have been present. I made a big assumption that the getOwnerIDCookie() simply wasn't returning the value fast enough, and because of this, I spent time moving around my async/await keywords with no success.

Long story short, the way I was looping over the cookies was the reason for my problems. I originally was using a foreach loop which was fine if the cookie I needed was last. If it was not the last cookie looped over, I was overwriting the OwnerID value previously found. I switched to a for loop, and used a break statement to bail out of the loop after I found the value I was looking for. I consider this a bug instead of an error since the code is working as written, but not what I intended.
https://github.com/ToddSpainhour/DoggyDetails/blob/main/DoggyDetails.ui/javascripts/helpers/data/OwnerIDData.js

Management Plan

Below is my management plan for the Doggy Details project. Looking back on the tasks I planned, I realized a needed some additions to the table below. These tasks revolved around the account creation and authentication parts of the project. This functionality was vital to the project yet was overlooked in my original planning. During development, these tasks took much more time than I anticipated, but this experience will help me make better plans and estimations in the future.

Green = Completed
Yellow = To-do

Phase 1: Database		**Time Estimation** (in Days)
Step 1.1	**Setup SQL Server**	**1**
Activity 1.1.1	Install SQL Server	1
Step 1.2	**Create database**	**7**
Activity 1.2.1	Create new database in SQL Server	1
Activity 1.2.2	Create User table	1
Activity 1.2.3	Create Pet table	1
Activity 1.2.4	Create Medicine table	1
Activity 1.2.5	Create Food table	1
Activity 1.2.6	Create Exercise table	1
Activity 1.2.7	Create Notes table	1

Phase 2: Source Control		**Time Estimation** (in Days)
Step 2.1	**Setup Source Control**	**1.5**
Activity 2.1.1	Download GitHub Desktop	**.5**
Activity 2.1.2	Create repo locally	**.5**
Activity 2.1.3	Push to GitHub	**.5**

Phase 3: REST APIs		**Time Estimation** (in Days)
Step 3.1	**Create solution in Visual Studio**	**1**
Activity 3.1.1	Connect to database	1
Step 3.2	**Endpoints for Owner**	**13**
Activity 3.2.1	Get all Owners	2
Activity 3.2.2	Create Owner	2
Activity 3.2.3	Edit Owner	2
Activity 3.2.4	Delete Owner	2
Activity 3.2.5	Check Username Availability	2
Activity 3.2.5	Login	2
Activity 3.2.6	GetAllPetsForThisOwner	1
Step 3.3	**Create endpoints for Pet**	**8**
Activity 3.3.1	Get all Pets for this OwnerID	2
Activity 3.3.2	Create Pet	2
Activity 3.3.3	Edit Pet	2
Activity 3.3.4	Delete Pet	2

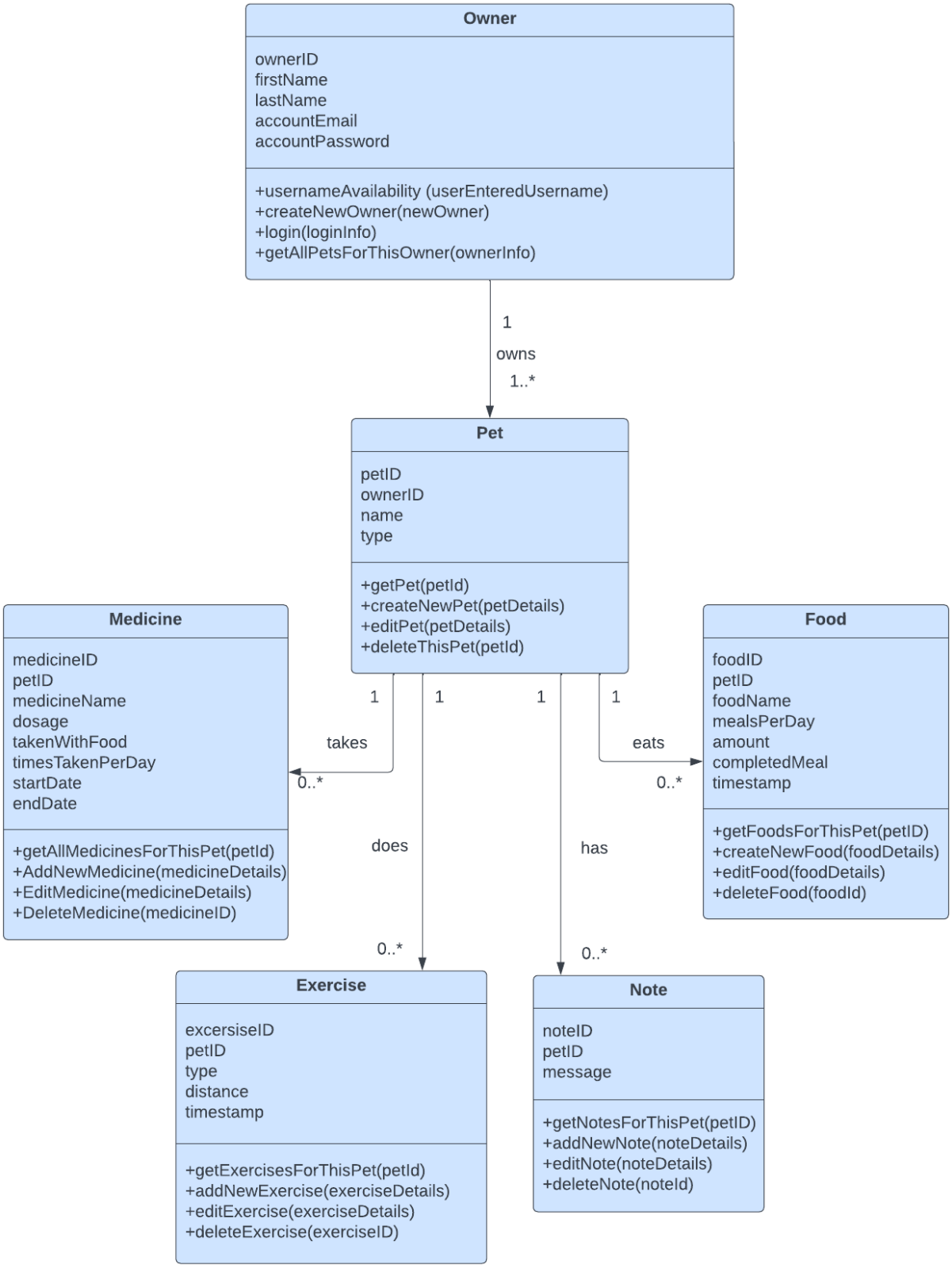
Step 3.4	**Create endpoints for Medicine**	**8**
Activity 3.4.1	Get all Medicine for Pet	2
Activity 3.4.2	Create Medicine for Pet	2
Activity 3.4.3	Edit Medicine for Pet	2
Activity 3.4.4	Delete Medicine for this pet	2
Step 3.5	**Create endpoints for Food**	**8**
Activity 3.5.1	Get all Food for this Pet	2
Activity 3.5.2	Create Food for this Pet	2
Activity 3.5.3	Edit Food for this Pet	2
Activity 3.5.4	Delete Food for this Pet	2
Step 3.6	**Create endpoint for Exercise**	**8**
Activity 3.6.1	Get all Exercise for this Pet	2
Activity 3.6.2	Create Exercise for Pet	2
Activity 3.6.3	Edit Exercise For this Pet	2
Activity 3.6.4	Delete Exercise for Pet	2
Step 3.7	**Create endpoints for Note**	**8**
Activity 3.7.1	Get all Notes for this Pet	2
Activity 3.7.2	Create Note for this Pet	2
Activity 3.7.3	Edit Notes for this Pet	2
Activity 3.7.4	Delete Note for this Pet	2

Phase 4: Frontend		**Time Estimation** (in Days)
Step 4.1	**Create frontend files**	**7**
Activity 4.1.1	Create index.html	1
Activity 4.1.2	Create login.html	3
Activity 4.1.3	Create dashboard.html	3
Step 4.2	**Create HTTP API Calls for Owner**	**5**
Activity 4.2.1	Get for Owner	1
Activity 4.2.2	Post for Owner	1
Activity 4.2.3	Put for Owner	1
Activity 4.2.4	Delete for Owner	1
Activity 4.2.5	Check Username Availability	1
Step 4.3	**Create HTTP API Calls for Pet**	**4**
Activity 4.3.1	Get for Pet	1
Activity 4.3.2	Post for Pet	1
Activity 4.3.3	Put for Pet	1
Activity 4.3.4	Delete for Pet	1
Step 4.4	**Create HTTP API Calls for Medicine**	**4**
Activity 4.4.1	Get for Medicine	1
Activity 4.4.2	Post for Medicine	1
Activity 4.4.3	Put for Medicine	1
Activity 4.4.4	Delete for Medicine	1
Step 4.5	**Create HTTP API Calls for Food**	**4**
Activity 4.5.1	Get for Food	1
Activity 4.5.2	Post for Food	1
Activity 4.5.3	Put for Food	1

Activity 4.5.4	Delete for Food	1
Step 4.6	**Create HTTP API Calls for Exercise**	**4**
Activity 4.6.1	Get for Exercise	1
Activity 4.6.2	Post for Exercise	1
Activity 4.6.3	Put for Exercise	1
Activity 4.6.4	Delete for Exercise	1
Step 4.7	**Create HTTP API Calls for Note**	**4**
Activity 4.7.1	Get for Note	1
Activity 4.7.2	Post for Note	1
Activity 4.7.3	Put for Note	1
Activity 4.7.4	Delete for Note	1
Step 4.8	**Style Pages**	**14**
Activity 4.8.1	Add CSS to pages	7
Activity 4.8.2	Make site mobile responsive	7

Database Entity Relationship Diagram

Below is the entity relationship diagram for the Doggy Details project.



High-Level Design – Layered Architecture

I chose to use the layered architecture design because of its simplicity. The flow of data between requests and responses is easier for me to understand, and this allows for quicker debugging. The design below was slightly adjusted from my original plan since I included the SQL queries in the endpoint code instead of putting them in their own repository file.