Pet Snatch

By Rita Lin (Rita-L), Mathew Liao (Marfield75), Nora Su (Nora-Su), Adeline Horstman (adho5467), David Thumma (Thumma-thunder), and Pradyumna Bangalore Sheshadri (prba4871)

Pet Snatch is a dynamic web application designed for buying and selling pets. Users can explore a variety of available pets from the homepage, accessing the pets information by simply clicking the “View Details” button, users may also search for a specific pet with Pet Snatch’s fully integrated search function. Pet Snatch enables users to register and sign in, allowing them to create their own profile, upload pets for purchase and purchase from other sellers.

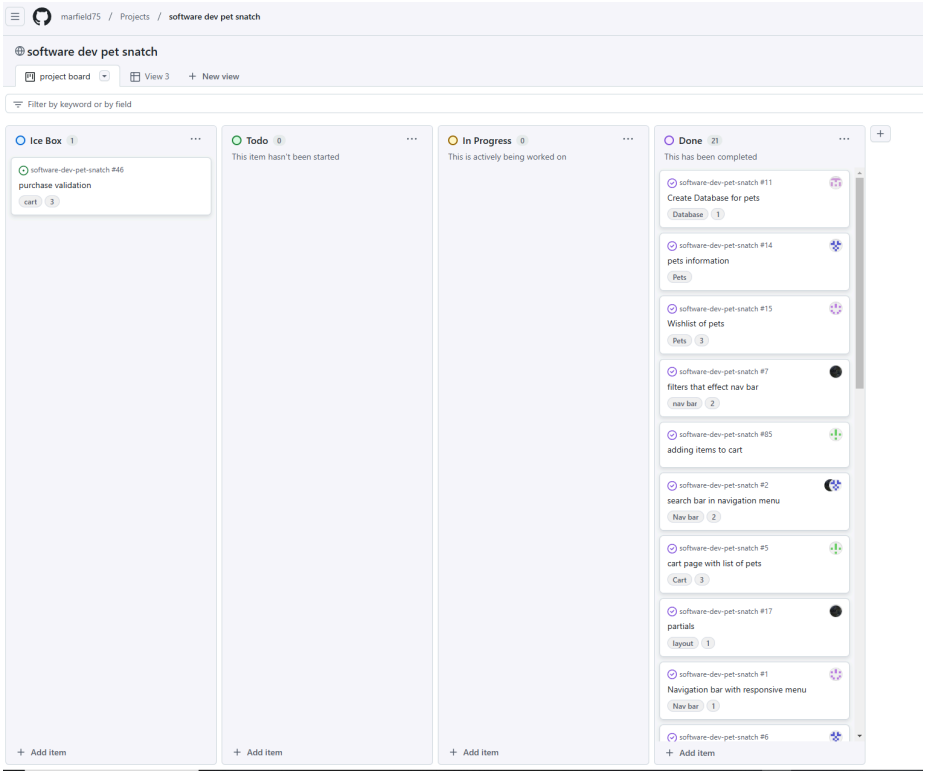
Once logged in users can manage their own pets and information from the profile page, where they can change their information and upload their own pets for purchase. Users can also add pets to their wishlist, allowing them to save pets for later purchase. Every pet listing includes a button to reroute the user to the pets own profile, which includes all necessary information about the pet such as breed, age, and price. Once on the pet’s profile, the user is able to purchase the pet by adding them to the cart and continuing to the purchase page to enter payment details.

Overall, Pet Snatch allows for an easy and seamless way to buy and sell pets for all pet owners or enthusiasts.

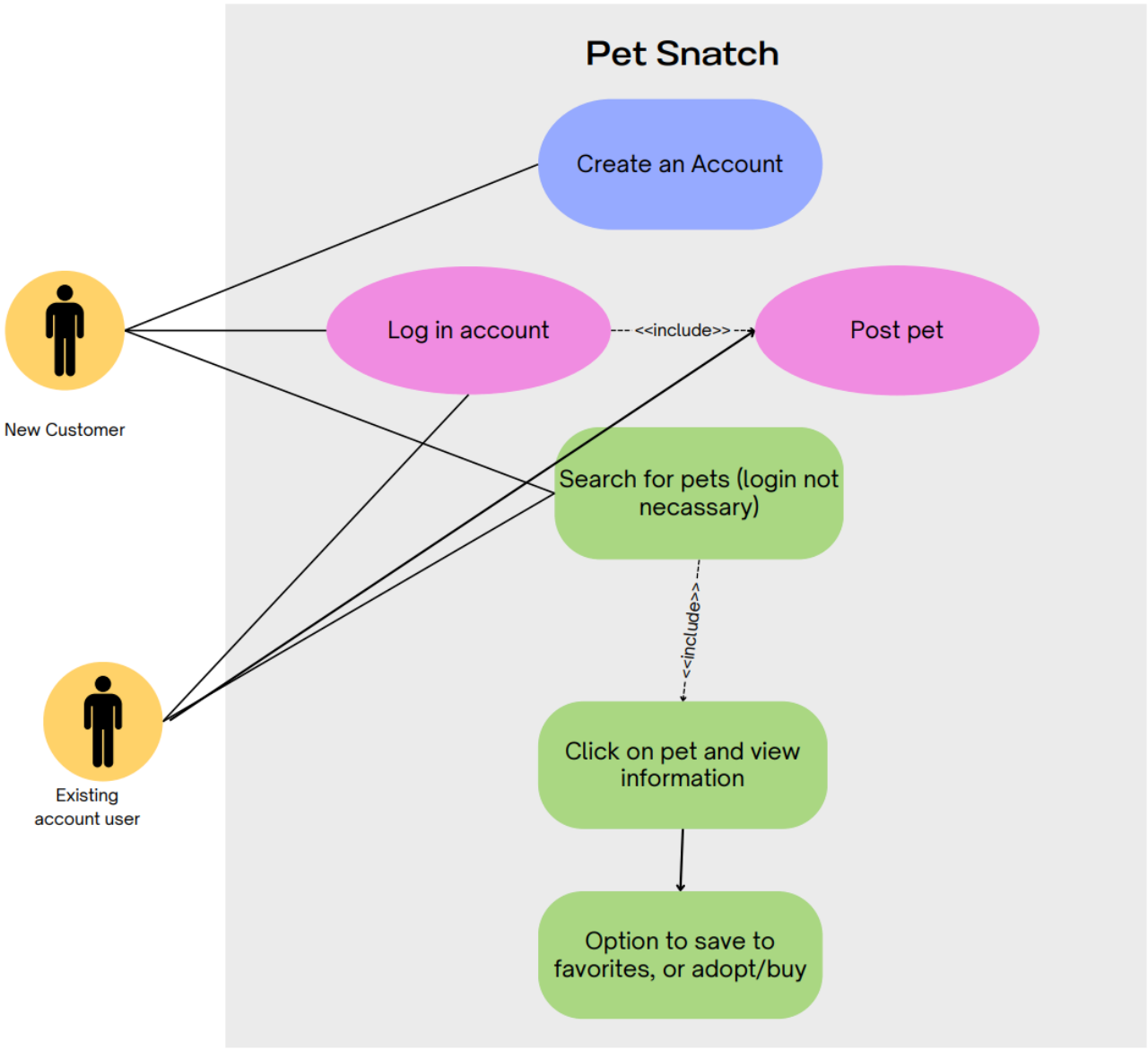
[Project Tracker]

[Link to GitHub Repository]

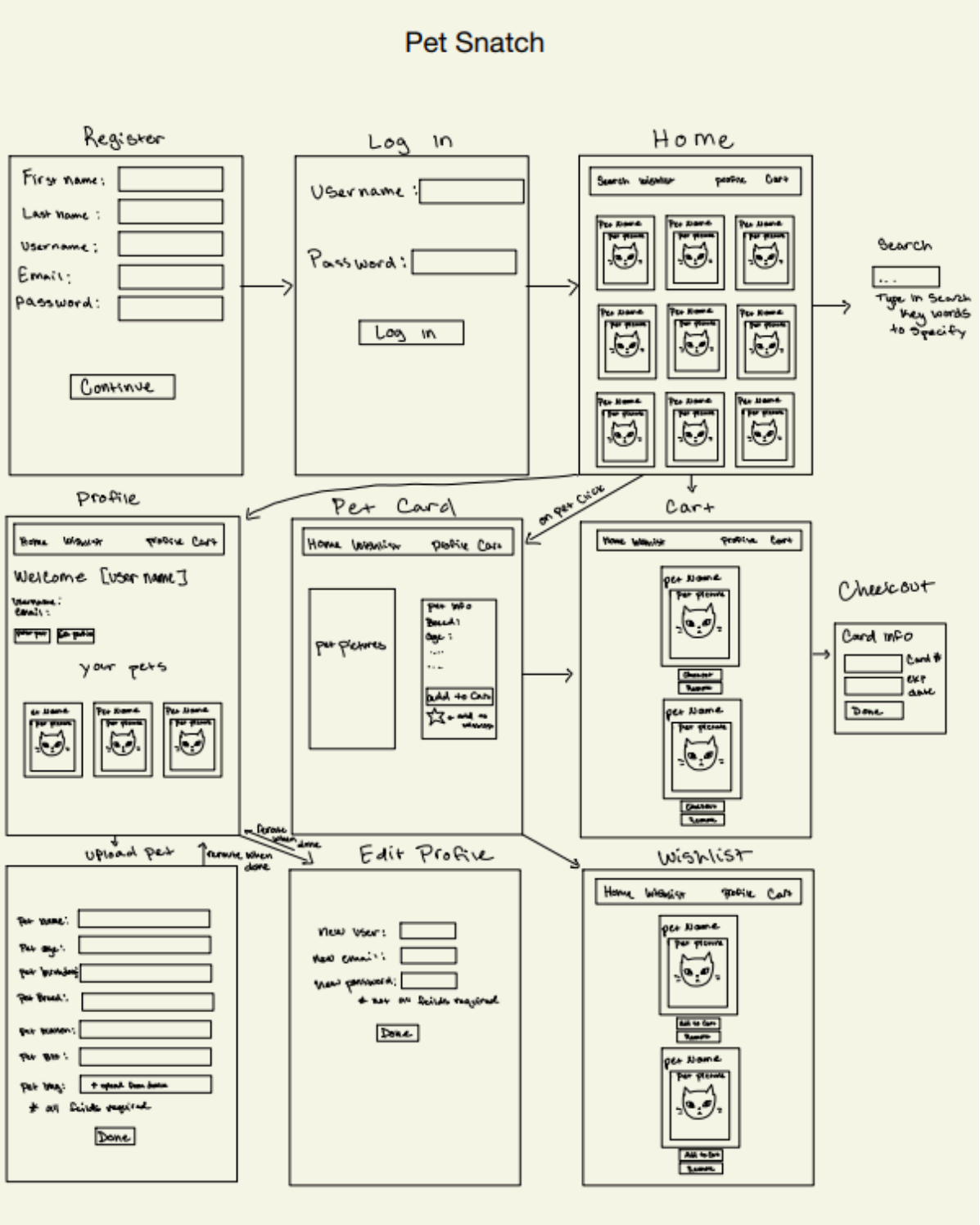
[Pet Snatch Demo Video]



Use-Case Diagram:

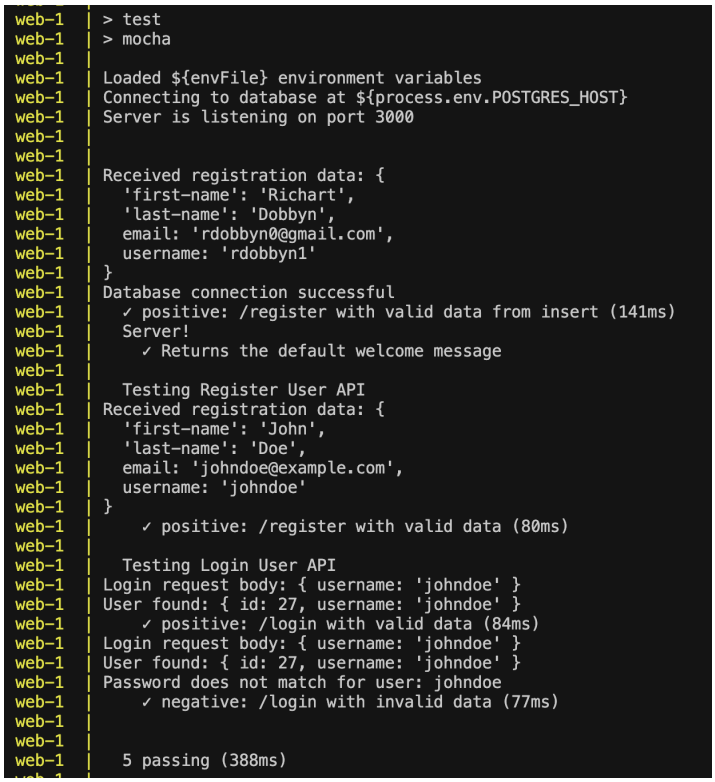


Wireframe Diagram:



UAT Plan:

1. A user should be able to visit a pet page while being signed in, and they should still be signed in while interacting with said pet page. Ex: Staying signed in while liking a pet photo. This will be tested on localhost using Chai.
    a. Instead of using Chai, we tested this manually: After signing in, the user can add any pet to their cart, add said pet to their wishlist, checkout with said pet, as well as logout.
2. A user can only register a pet when signed in, including all the features when adding a pet, such as its attributes. Ex: Someone logged out shouldn't be able to register a pet, they should be redirected to the login page. This will be tested on localhost using Chai.
    a. Instead of using Chai, we tested this manually: The register-a-pet option doesn't appear unless you're signed in, as you need to access your profile page, which is "locked" until you register an account and sign in. After signing in, when going to your profile, you now have the option to register a pet, which includes all of its attributes.
3. A user can add pets to their favorites list only when signed in, and when they sign out, the favorites list should be saved somewhere, so that when they sign back in the information is saved. Ex: Someone adds a couple of pets to their favorites list, then logs out and when they log back in later, they should still have their favorites list, untouched. This will be tested on localhost using Chai.
    a. Instead of using Chai, we tested this manually: A user cannot favorite a pet until they are signed in. After signing in, a user can favorite a pet, and then even after logging out, the favorited pet is saved when they log back in.



```
web-1  | > test
web-1  | > mocha
web-1  |
web-1  | Loaded ${envFile} environment variables
web-1  | Connecting to database at ${process.env.POSTGRES_HOST}
web-1  | Server is listening on port 3000
web-1  |
web-1  |
web-1  | Received registration data: {
web-1  |   'first-name': 'Richart',
web-1  |   'last-name': 'Dobbyn',
web-1  |   email: 'rdobbyn0@gmail.com',
web-1  |   username: 'rdobbyn1'
web-1  | }
web-1  | Database connection successful
web-1  |   ✓ positive: /register with valid data from insert (141ms)
web-1  |   Server!
web-1  |     ✓ Returns the default welcome message
web-1  |
web-1  |   Testing Register User API
web-1  | Received registration data: {
web-1  |   'first-name': 'John',
web-1  |   'last-name': 'Doe',
web-1  |   email: 'johndoe@example.com',
web-1  |   username: 'johndoe'
web-1  | }
web-1  |     ✓ positive: /register with valid data (80ms)
web-1  |
web-1  |   Testing Login User API
web-1  | Login request body: { username: 'johndoe' }
web-1  | User found: { id: 27, username: 'johndoe' }
web-1  |     ✓ positive: /login with valid data (84ms)
web-1  | Login request body: { username: 'johndoe' }
web-1  | User found: { id: 27, username: 'johndoe' }
web-1  | Password does not match for user: johndoe
web-1  |     ✓ negative: /login with invalid data (77ms)
web-1  |
web-1  |
web-1  |   5 passing (388ms)
```

Contributions:

Nora Su: I mainly worked on the individual pet pages and the cart page. I used HTML and CSS to design the layout of the pet pages that can be accessed from the homepage, which included the pet's attributes, a button to add the pet to the cart, and a button to add the pet to the user's favorites. I then wrote API routes to retrieve pet data and used Handlebars to display the relevant information. I designed the cart page similarly. I also recorded all of the TA meeting logs and I created the slides for the group presentation during recitation.

Rita: In this project, I contributed to both front-end and back-end development. On the front-end, I set up the login and registration pages, created a dropdown menu, payment page, and designed a custom logo, integrating it into the navbar. I also sourced and uploaded most of the project's image resources. This required skills in HTML, CSS, JavaScript, and responsive design. On the back-end, I developed API routes for "add to cart" functionality using Node.js and Express.js, ensuring robust data validation and error handling. Additionally, I fixed the database to align with the image resources.

Adeline Horstman: I worked on the profile and edit profile pages as well as all the API routes involved with those pages. I used HTML with inline styling to create my pages and Javascript to create the API routes necessary to allow the user to navigate to my pages, as well as making a function to grab the user data needed to display the profile page for the current user, and update their information on the edit user page. As well as creating the project report.

David Thumma: I mainly worked on the endpoints using Node.js and I worked on displaying pet information on the front end using handlebars as well. I wrote some of the routes and set up Multer in Node.js which was middleware we used to allow image uploads. I also implemented the search and checkout feature.

Matthew Liao: I created the Github repository, worked on implementing the home page, and implemented the HTML, CSS, Handlebars, and API routes to display the pet cards. I debugged issues with the home page and modified the layout of the pet cards. I modified data within our database to match conventions and added documentation for the UATs and testing. I implemented lab 13, initialized Render, and deployed our project. After initializing Render, I helped debug it and fixed some issues it was causing with Docker.

Pradyumna Bangalore Sheshadri: I initially worked on creating the database tables and inserting random data into it using create.sql and insert.sql. Then, I worked on the security of the password when registering and logging in. I did this by a mechanism called hashing. I also worked on inputting the pet image from the user in registering the pet. We, as a team, decided that, instead of having a logout page, we will just redirect the user to the home page with a message when the user logs out. I worked on the endpoint and message displaying for this.

Deployment: Our deployment method is through Render which can be found in the [Render Pet Snatch Link]