The tools I used for testing included web page inspection, Redux Devtools, and Postman. Initially I was working on both front-end and back-end design. As part of the learning process and seeing how elements interact with each other, a visual inspection would be the first step in determining if react code was performing as inspected. After a visual test, inspecting the page to look at console output and network usage was a helpful tool. I could check the value of elements I was working on by printing them to the console. During sprint 2 I shifted my focus to the backend server. I started working with the Django Rest Framework and needed to test integration of requests to the backend server. Postman was useful for creating endpoints that could test the backend without having to work with React components. Once I was confident that an endpoint was returning data I expected it to I would try working with axios to retrieve the data from the backend server. Some issues I had to deal with were null values being retrieved, failed authentication, incorrect request addresses, as well as other issues I needed to debug.

Postman became a useful tool for creating users and testing API endpoints. I created multiple unique users that I could register, login, and edit profile information. This allowed me to create profiles from an empty database quickly and check changes between multiple users. By tracking output between the browser console, postman output, and the Django servers responses, including errors and python print statements I could track requests and element information across the site. In the future I would want to add automatic testing of running different collections of endpoints. There were times that code changes would affect the output of different endpoints and it was difficult to identify when the change happened or how long it had been since the system changed. Having a simple command I could run to double check that endpoints were working as expected would have saved me time looking for issues caused by working in different github branches. Taking a test driven development approach would be especially helpful when working with creating a backend server. It would allow me to work in isolation on the backend and work to get each test to a passing point.