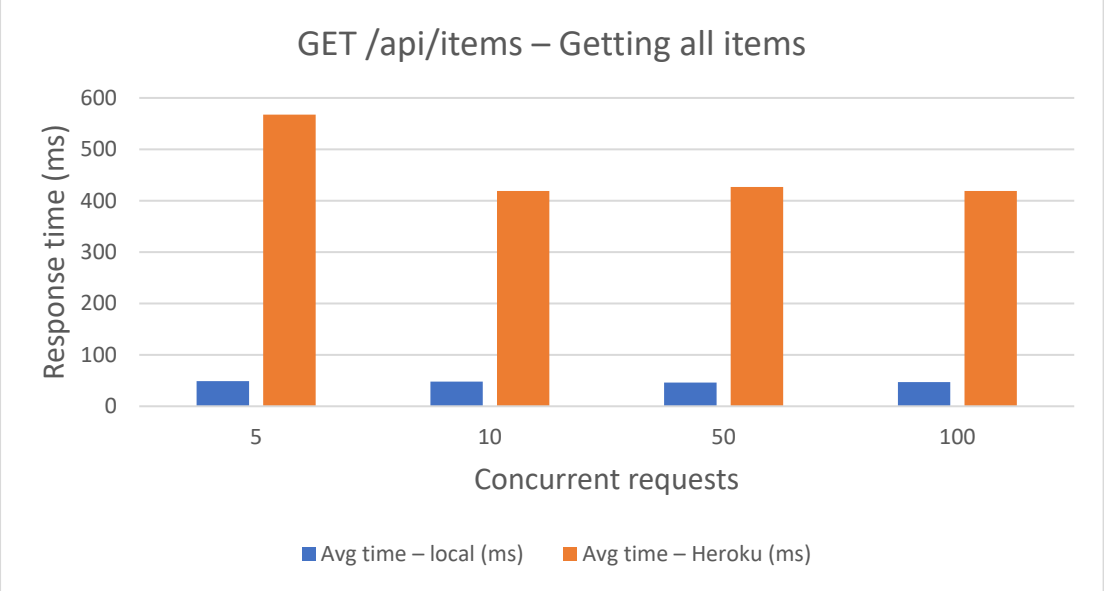
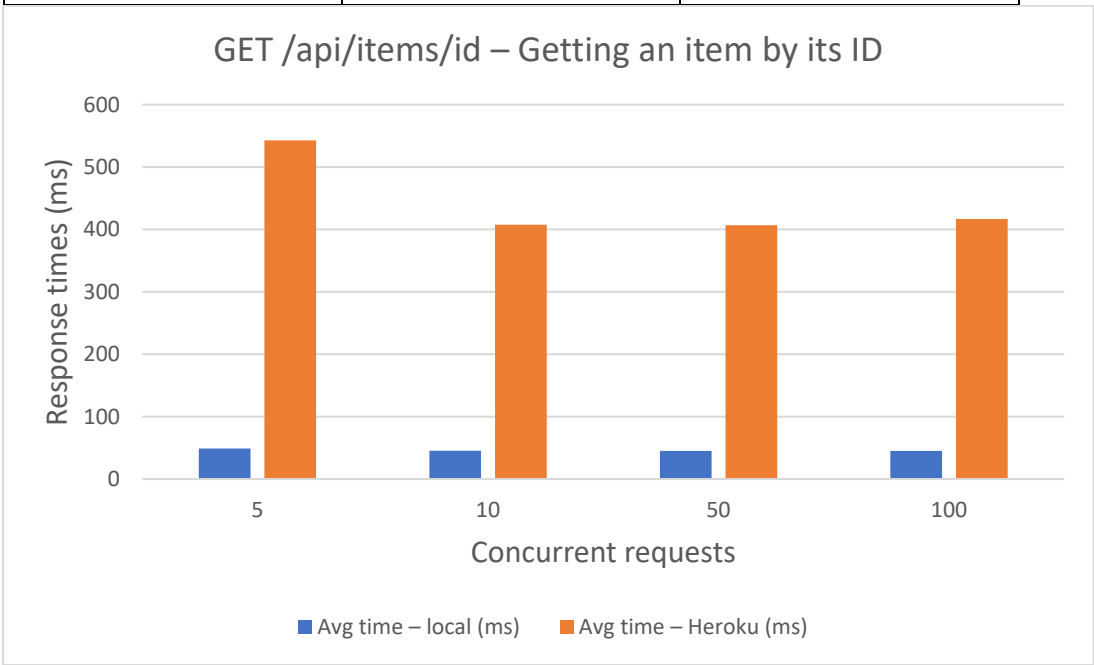
GET /api/items – Getting all items

Concurrent requests	Avg time – local (ms)	Avg time – Heroku (ms)
5	49	567.6
10	48	419.0
50	46	426.8
100	47	418.9



GET /api/items/id – Getting an item by its ID

Concurrent requests	Avg time – local (ms)	Avg time – Heroku (ms)
5	49.0	542.6
10	45.4	407.5
50	45.1	406.8
100	45.0	416.7
1000	44.8	

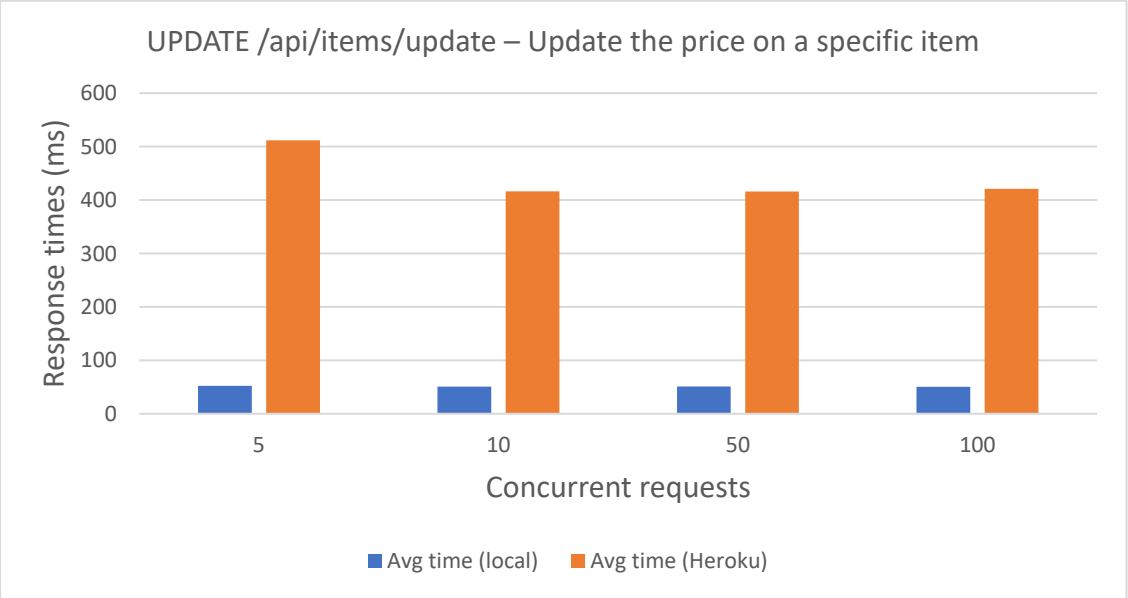


GET /api/items/10 – Getting 10 random items

Concurrent requests	Avg time (local)	Avg time (Heroku)
5	47.2	419.4
10	47.7	417.1
50	46.3	425.66
100	45.5	419.8

UPDATE /api/items/update – Update the price on a specific item

Concurrent requests	Avg time (local)	Avg time (Heroku)
5	52.2	511.8
10	50.8	416.2
50	51.0	416
100	50.5	421.12



Create requests.

CREATE /local-signup – Create a new user (Override matching username check)

Concurrent requests	Avg time (local)	Avg time (Heroku)
5	92.2	
10	89.9	
50	90.04	
100	89.45	

Section d - Performance Evaluation.

For this project we used MongoDB as our database. This is connected to and interacted with using our RESTful web application services. For the most part, as can be seen in the performance charts we developed using localhost with little to no wait time from the database. Items would already be loaded when the page loads.

When testing the performance with concurrent users connecting to the database, there was little to no bottlenecking. Each user had roughly the same response time. This was pushed the most when running over 100 requests, at which point it became so long in some cases there was no time to finish the testing. But this was rare and most experienced fine connections.

One clear drawback to performance is that due to us using a free version of MongoDB, the database uses a shared cluster. This means it scales horizontally over multiple servers with sharding. As opposed to a replica database which would ensure higher availability and better performance.

The database can be expected to cause minor bottlenecking. It was unnoticeable during our development and performed well during our testing. If the need were to ever arise that better performance was required, there are options within MongoDB and elsewhere which would allow that to be seamless.