JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

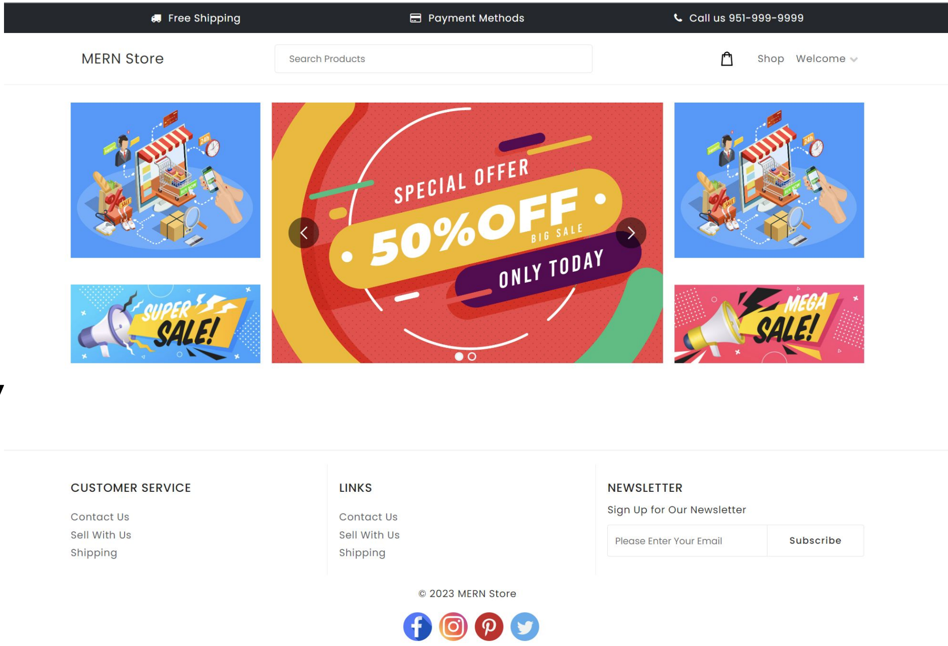
MERN ECommerce App Test

Kai Ko, Boyuan Wang, Qitong Niu

Software Testing and Debugging

Introduction

- This app is a ecommerce web-app created in the MERN (MongoDB, Express, React, Node) stack.
- The app allows for three main users
  - Buyers to view store, products, category, brands and purchase them
  - Sellers and merchants can manage their products, and brands
  - Admins can manage everything that is in the store
- The backend is provided by Node, with a MongoDB database and express middleware, where as the frontend is created with React Redux.



Our Testing

- We performed 3 different types of tests
  - Unit & Integration Testing (API Testing mainly)
    - Black box
    - White box
    - Mock in order to simulate errors being thrown, and email service
  - Load Testing
  - UI Testing
- Goal:
  - Achieve as close to 100% branch coverage as possible with the APIs
  - Conduct load testing
  - UI test the sign-in/out, cart, and product features

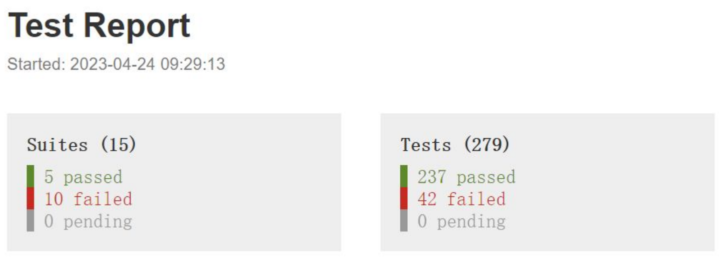
Tools Used

- Unit & Integration Testing
  - Jest
  - Supertest
- Load Testing
  - Artillery
- UI Testing
  - Jest
  - Cypress

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Live Demo

Coverage Chart

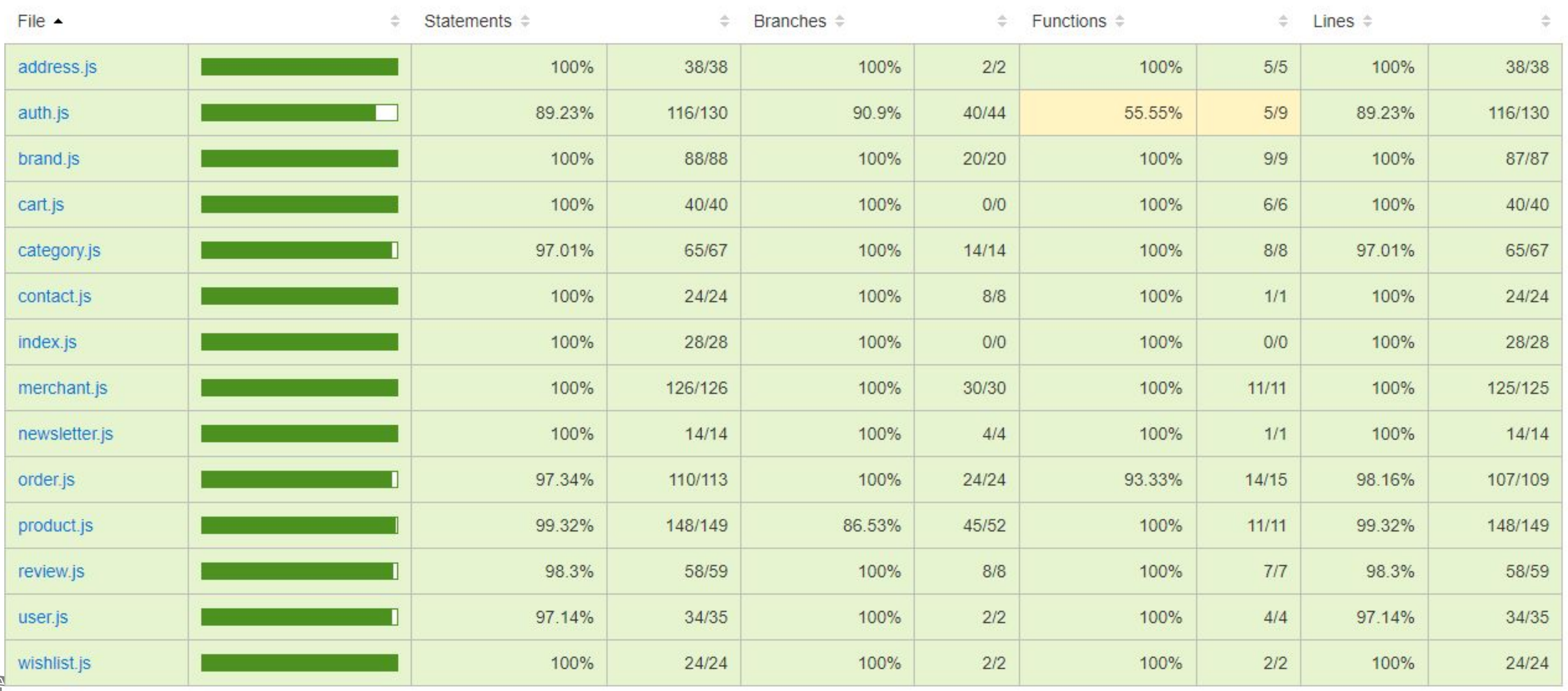


Test Report

Started: 2023-04-24 09:29:13

Suites (15)
5 passed
10 failed
0 pending

Tests (279)
237 passed
42 failed
0 pending



File ▲		Statements		Branches		Functions		Lines	
address.js		100%	38/38	100%	2/2	100%	5/5	100%	38/38
auth.js		89.23%	116/130	90.9%	40/44	55.55%	5/9	89.23%	116/130
brand.js		100%	88/88	100%	20/20	100%	9/9	100%	87/87
cart.js		100%	40/40	100%	0/0	100%	6/6	100%	40/40
category.js		97.01%	65/67	100%	14/14	100%	8/8	97.01%	65/67
contact.js		100%	24/24	100%	8/8	100%	1/1	100%	24/24
index.js		100%	28/28	100%	0/0	100%	0/0	100%	28/28
merchant.js		100%	126/126	100%	30/30	100%	11/11	100%	125/125
newsletter.js		100%	14/14	100%	4/4	100%	1/1	100%	14/14
order.js		97.34%	110/113	100%	24/24	93.33%	14/15	98.16%	107/109
product.js		99.32%	148/149	86.53%	45/52	100%	11/11	99.32%	148/149
review.js		98.3%	58/59	100%	8/8	100%	7/7	98.3%	58/59
user.js		97.14%	34/35	100%	2/2	100%	4/4	97.14%	34/35
wishlist.js		100%	24/24	100%	2/2	100%	2/2	100%	24/24

Bugs Found

Main Faults:

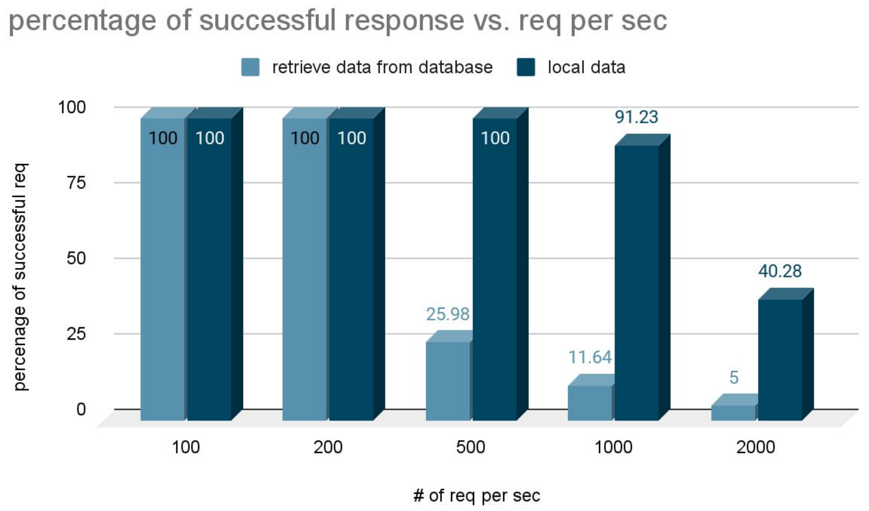
1. No Validators in Backend (Almost every api)

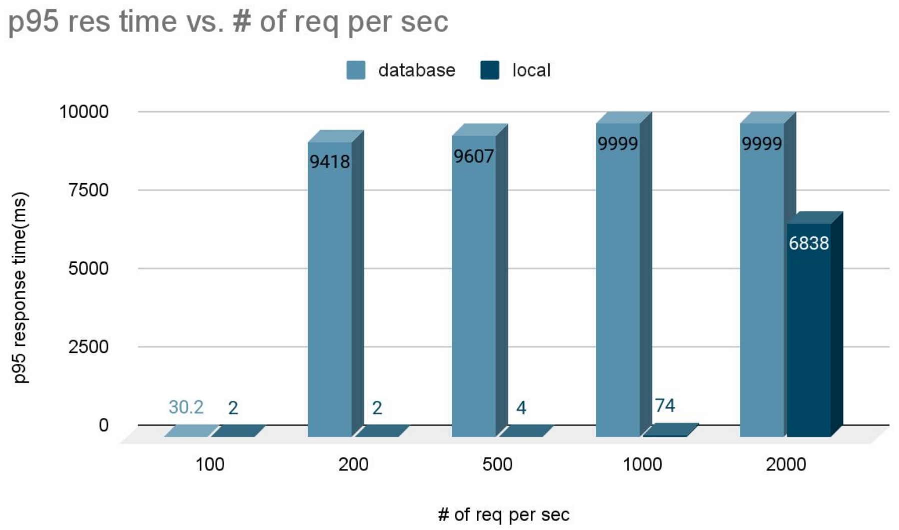
   e.g. if login request’s email is an array, it just processes it (find one that match one of the emails).

2. No Authorization Checking in some APIs

3. Other faults…
   a. Responses aren’t returned in API causing faults when there are other responses after
   b. One place where <= should have been used instead of < 0

Load Testing Chart





Conclusion

- Break down an industry level app and perform comprehensive test
- With minimal documentation from the creators of the app, we were able to achieve our goal of 100% branch coverage in terms of code that was reachable with no faults in addition to load testing and UI testing.

- Lessons learnt:
  - Old code bases have a lot of dependency issues

Any Questions?

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING