



## Etsy Prototype - Lab 2

Snigdha Chaturvedi | 015957435

Youtube Demo: <https://www.youtube.com/watch?v=-Wn5MniiLVU>  
Git Repository: <https://github.com/Snichat97/EtsyPrivateAWS>

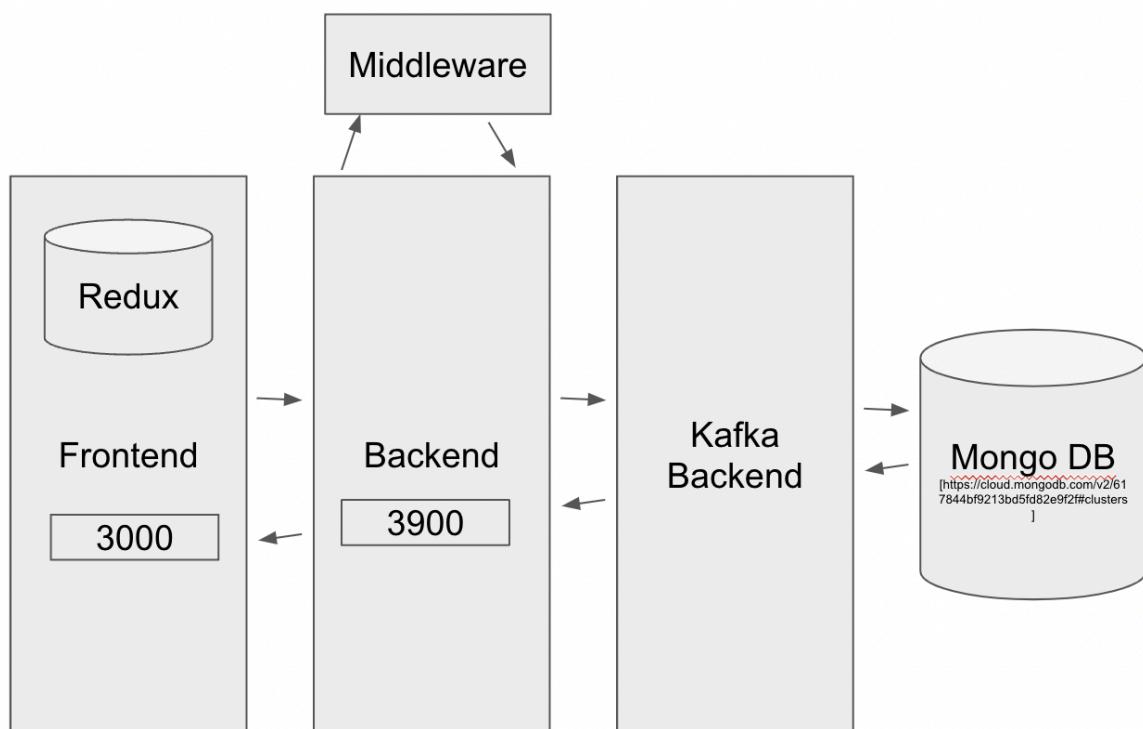
## Goals and Purpose:

My goals from this project are to design and deploy an end-to-end application and explore in-depth backend API construction, various deployment strategies, and using the Redux library for state management across multiple components. Further, in Lab 2, I have explored the Kafka messaging queue between the backend and the database which enables handling a high number of requests. Also migrating the database from MySQL to MongoDB helped me to identify the differences between NoSQL and Relational Databases and understand in which use case which database can be leveraged. The system is a prototype of the “Etsy” Application which allows users to buy products and host their own shops.

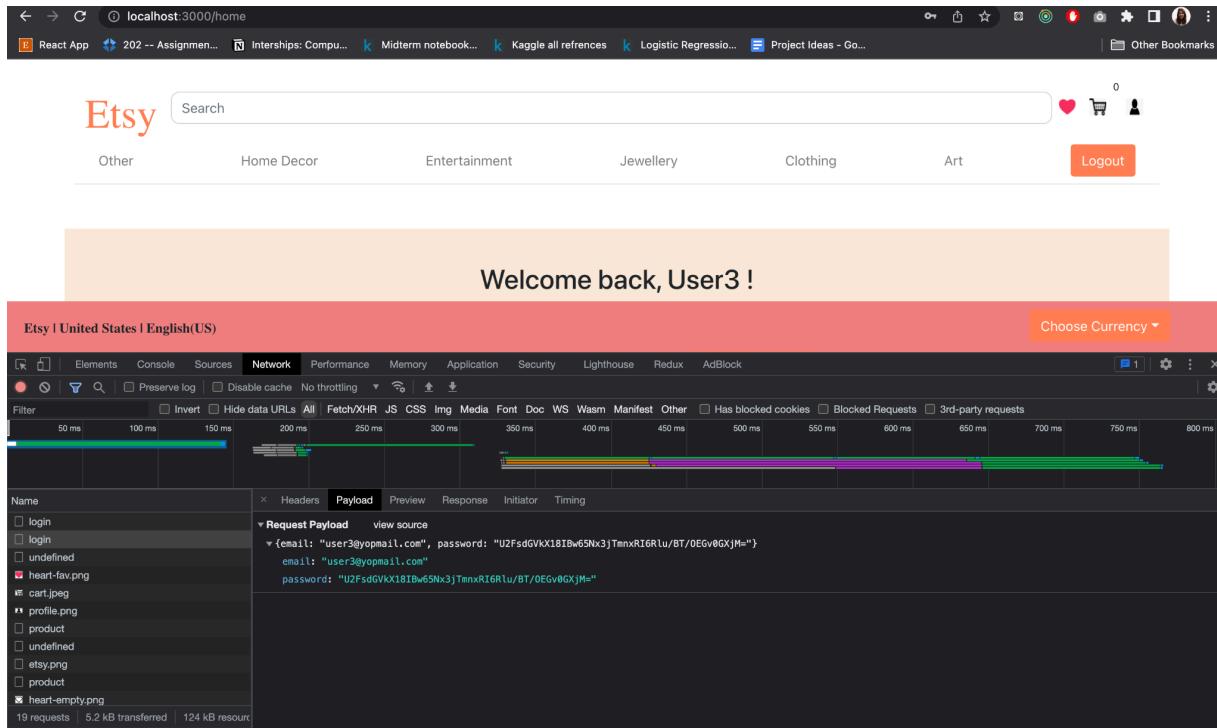
## Salient features of the System:

The application uses ReactJS for the frontend and NodeJs running on Express Server for the backend where the call is sent to the middleware to test the authentication token. On verification, it is sent to the kafka-backend.

An AWS EC2 instance holds the complete application. On port 3000 the frontend of the application is hosted and on 3900 the backend is running. Using the mongoose library it connects to the MongoDB database. The MongoDB server is running on Atlas (<https://cloud.mongodb.com/>)



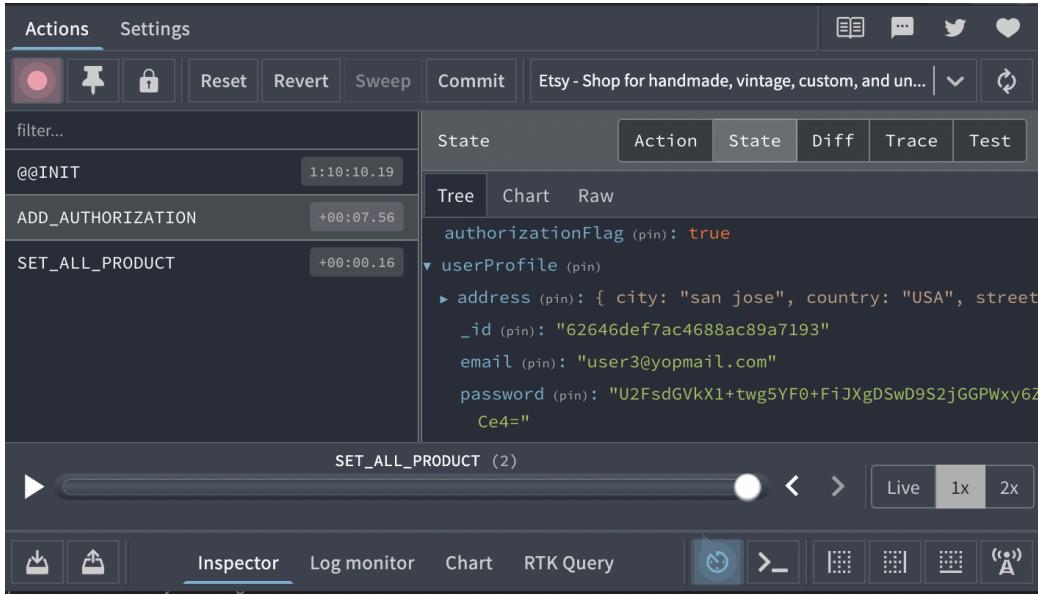
# Screen Previews



**On the Login Call, the Authorization token is returned**



## Storing the User Profile in Redux



## Gift Wrapping and Gift messaging option

The screenshot shows the Etsy website with a shopping cart open. The cart contains three items:

- Chalk Board**: Quantity: 1, Price: 35. Gift Wrap is checked, and a message "Happy Birthday , Snigdha!" is entered in the message field. There are +, -, and Delete buttons.
- Pencil Holder**: Quantity: 1, Price: 30. Gift Wrap is unchecked.
- White Board**: Quantity: 1, Price: 20. Gift Wrap is unchecked.

The total price is listed as **85**. At the bottom, there is a large orange button labeled **Complete Purchase**.

At the bottom of the page, there are language and currency selection dropdowns: English(US) and Choose Currency ▾.

## Order Pagination

The screenshot shows a user interface for managing orders. At the top, there is a navigation bar with links for Other, Home Decor, Entertainment, Jewellery, Clothing, Art, and a Logout button. Below the navigation bar is a section titled "Order Listing" with a sub-section "Number of Orders". A dropdown menu is open, showing the value "2". Below this, there are two order entries:

**Order Number : 62646e077ac4688ac89a7197**

**Pencil Holder**  
Quantity : 1  
Price : 30  
Shop Name : 626469f87ac4688ac89a714d

**Order Number : 62646e177ac4688ac89a719d**

**Table Lamp**  
Quantity : 3  
Price : 90  
Shop Name : 626469f87ac4688ac89a714d

**Chalk Board**  
Quantity : 2  
Price : 70  
Shop Name : 626469f87ac4688ac89a714d

At the bottom of the list is a pagination control with buttons for navigating between pages.

Etsy | United States | English(US)      Choose Currency ▾

## Order History - With Gift Wrap, Customized message, and Order Pagination

The screenshot shows an order history page for order number 6265075c440f693f5e505db3. The page displays three items:

**Chalk Board**  
Quantity : 1  
Price : 35  
Shop Name : 626469f87ac4688ac89a714d  
**Gift Product | Message :Happy Birthday, Snigdha !**

**Pencil Holder**  
Quantity : 1  
Price : 30  
Shop Name : 626469f87ac4688ac89a714d

**White Board**  
Quantity : 1  
Price : 20  
Shop Name : 626469f87ac4688ac89a714d

At the bottom of the list is a pagination control with buttons for navigating between pages.

# MongoDB - Deployed on Atlas

The screenshot shows the MongoDB Atlas interface. On the left, a sidebar navigation includes 'Project 0', 'Atlas', and 'Realm'. Under 'Database', there's a 'myFirstDatabase' section with 'orders', 'products', 'shops', and 'users' collections. The main panel displays the 'orders' collection with a document preview. The document contains fields like 'iduser', 'productItems', and 'giftMessage'. A 'FILTER' bar at the top allows filtering by field value. Below the preview, a list of documents is shown, each with a preview icon and a delete button. At the bottom, a status bar indicates 'All Good'.

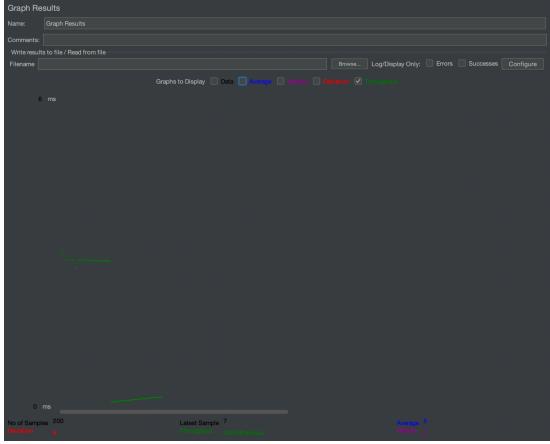
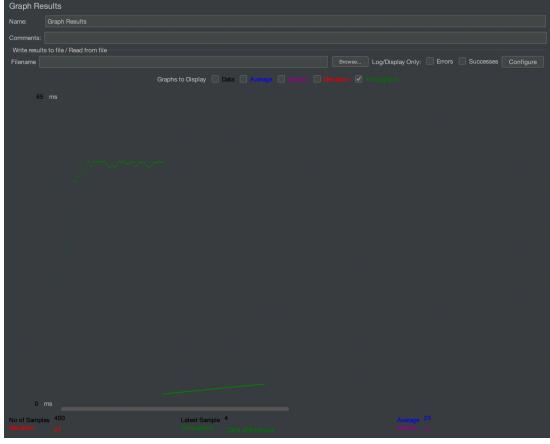
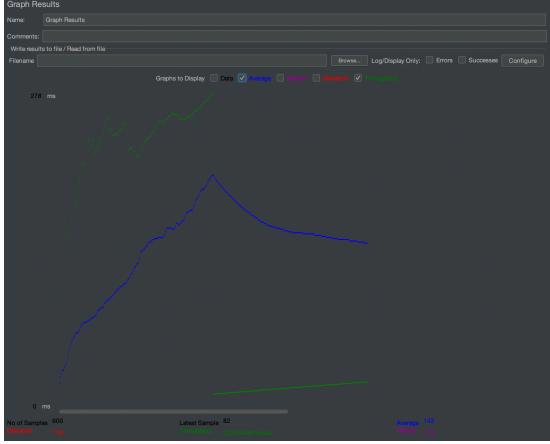
# Images deployed on S3

The screenshot shows the AWS S3 console. The left sidebar includes 'Amazon S3', 'Buckets', 'Storage Lens', and 'AWS Marketplace for S3'. The main area shows the 'etsybucketaws' bucket with 14 objects listed. The objects are:

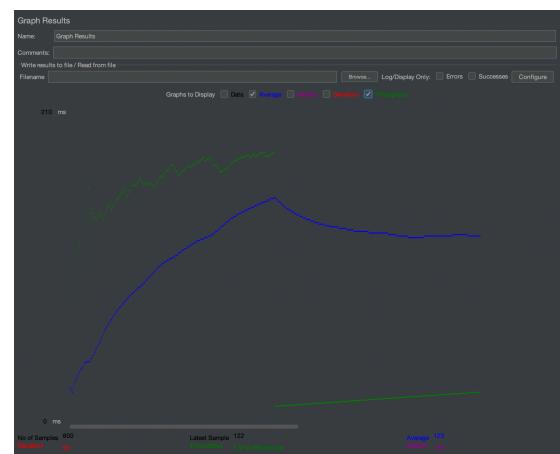
Name	Type	Last modified	Size	Storage class
02e26fd24430e283ab722e1beb532108	-	April 23, 2022, 14:09:20 (UTC-07:00)	8.5 KB	Standard
13eaf8f729b0da7925885d6bf69532	-	April 23, 2022, 19:58:16 (UTC-07:00)	8.2 KB	Standard
1b5d20a045830a2358b5c521e97350	-	April 23, 2022, 14:20:07 (UTC-07:00)	8.5 KB	Standard
354ee5fb7ab7b1e7ac0d30ec725a2589	-	April 23, 2022, 14:09:51 (UTC-07:00)	4.9 KB	Standard
3bf58bb3d76d4fb04ebd3531186cc62a	-	April 23, 2022, 14:11:23 (UTC-07:00)	6.1 KB	Standard
4116603c375a7190f0821e1d249e771	-	April 23, 2022, 20:11:39 (UTC-07:00)	8.2 KB	Standard
9d5ff1abf7eb2dc7134ef5922dca5c	-	April 23, 2022, 14:19:33 (UTC-07:00)	7.1 KB	Standard
aab14fe6c0a42be5d90193c20edcf	-	April 23, 2022, 14:10:18 (UTC-07:00)	9.3 KB	Standard
bef7f7c790fddab5565adcf7d76821b5e	-	April 23, 2022, 19:58:35 (UTC-07:00)	8.2 KB	Standard
bfb137901d89744dec01874bf732b51b	-	April 23, 2022, 14:21:26 (UTC-07:00)	7.2 KB	Standard
c7ba5ff46a7da8f72ccb5412864f67	-	April 23, 2022, 14:20:37 (UTC-07:00)	9.5 KB	Standard
d001429fc4d64a73676745164147a	-	April 23, 2022, 14:01:32 (UTC-07:00)	3.3 MB	Standard
feesee61b9fb03d4d5a40d7ca76f571	-	April 23, 2022, 14:19:01 (UTC-07:00)	9.1 KB	Standard
fef1ebf660782b0bdbee940d653194	-	April 23, 2022, 20:13:34 (UTC-07:00)	8.2 KB	Standard

# Testing

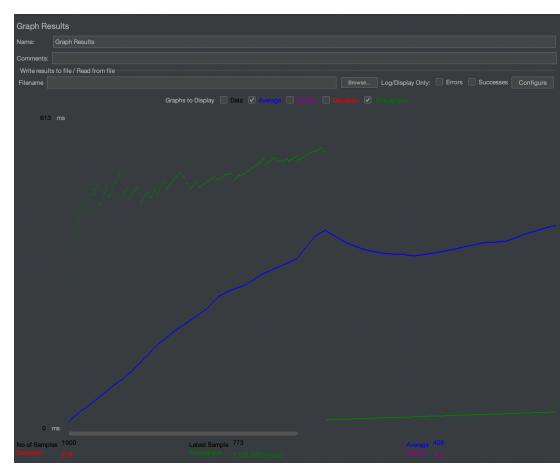
## 1. JMeter Testing

Number of concurrent users	Graph pooling vs single connection
100 Users	 <p>Graph Results Name: Graph Results Comments: Write results to file / Read from file Filename: <input type="text"/> Browse... LogDisplay Only: <input type="checkbox"/> Errors: <input type="checkbox"/> Successes: <input checked="" type="checkbox"/> Throughput Graphs to Display: <input type="checkbox"/> Data <input checked="" type="checkbox"/> Average <input type="checkbox"/> Min <input type="checkbox"/> Max <input type="checkbox"/> Deviation <input checked="" type="checkbox"/> Throughput 0 ms Latest Sample: 7 ms Throughput: 140 samples/sec No of Samples: 200 Duration: 0 ms Average: 5 ms Median: 3 ms</p>
200 Users	 <p>Graph Results Name: Graph Results Comments: Write results to file / Read from file Filename: <input type="text"/> Browse... LogDisplay Only: <input type="checkbox"/> Errors: <input type="checkbox"/> Successes: <input checked="" type="checkbox"/> Throughput Graphs to Display: <input type="checkbox"/> Data <input type="checkbox"/> Average <input type="checkbox"/> Min <input type="checkbox"/> Max <input type="checkbox"/> Deviation <input checked="" type="checkbox"/> Throughput 0 ms Latest Sample: 4 ms Throughput: 130 samples/sec No of Samples: 400 Duration: 37 ms Average: 10 ms Median: 11 ms</p>
300 Users	 <p>Graph Results Name: Graph Results Comments: Write results to file / Read from file Filename: <input type="text"/> Browse... LogDisplay Only: <input type="checkbox"/> Errors: <input type="checkbox"/> Successes: <input checked="" type="checkbox"/> Throughput Graphs to Display: <input type="checkbox"/> Data <input checked="" type="checkbox"/> Average <input type="checkbox"/> Min <input type="checkbox"/> Max <input type="checkbox"/> Deviation <input checked="" type="checkbox"/> Throughput 0 ms Latest Sample: 82 ms Throughput: 100 samples/sec No of Samples: 600 Duration: 110 ms Average: 140 ms Median: 115 ms</p>

## 400 Users



## 500 Users



## 2. Mocha Testing

```
JS db.js JS test.js X ProductItem.jsx U # App.css M Login.jsx U JS query.js JS user.js .../routes JS user.js .../n ...  
Lab1 > backend > test > JS test.js > [e] apiUrl  
1  const axios = require("axios");  
2  
3  const assert = require("assert");  
4  
5  const apiUrl = "http://localhost:3900/api";  
6  
7  describe("POST - check for sign in", () => {  
8    it("/user/login", (done) => {  
9      axios  
10        .post(apiUrl + "/user/login", { email: "snigdha.chaturvedi@sjtu.edu", password: "snigdha" })  
11        .then((response) => {  
12          console.log(response.data);  
13          assert.equal(response.status, 200);  
14          done();  
15        })  
16        .catch((err) => {  
17          done(err);  
18        });  
19    });
20  });
21  
22  describe("GET - get customer details by id", () => {  
23    it("/user/25", (done) => {  
24      axios  
25        .get(apiUrl + "/user/25")  
26        .then((response) => {  
27          console.log(response.data);  
28          assert.equal(response.status, 200);  
29          done();  
30        })  
31        .catch((err) => {  
32          done(err);  
33        });
34    });
35  });
36  
37  describe("GET - product details", () => {  
38    it("/product/14", (done) => {  
39      axios  
40        .get(apiUrl + "/product/14")  
41        .then((response) => {  
42          console.log(response.data);  
43          assert.equal(response.status, 200);  
44          done();  
45        })  
46        .catch((err) => {  
47          done(err);  
48        });
49    });
50  });
51  
52  describe("GET - shop details", () => {  
53    it("/shop/16", (done) => {  
54      axios  
55        .get(apiUrl + "/shop/16")  
56        .then((response) => {  
57          console.log(response.data);  
58          assert.equal(response.status, 200);  
59          done();  
60        })  
61        .catch((err) => {  
62          done(err);  
63        });
64    });
65  });
66  
67  describe("GET - user's shop ", () => {  
68    it("/user/25", (done) => {  
69      axios  
70        .get(apiUrl + "/user/25")  
71        .then((response) => {  
72          console.log(response.data);  
73          assert.equal(response.status, 200);  
74          done();  
75        })  
76        .catch((err) => {  
77          done(err);  
78        });
79    });
80  });
81
```

### 3. React Testing

Component/Page	React Testing Library
<b>Shop Page</b>	<pre>import React from "react"; import { render, screen } from "@testing-library/react"; import AddShopImage from "../atom/AddShopImage.jsx"; import { BrowserRouter } from "react-router-dom"; import { Provider } from "react-redux"; import store from "../redux/store";  test("renders Shop Page", () =&gt; {   render(     &lt;Provider store={store}&gt;       &lt;React.StrictMode&gt;         &lt;BrowserRouter&gt;           &lt;AddShopImage /&gt;         &lt;/BrowserRouter&gt;       &lt;/React.StrictMode&gt;     &lt;/Provider&gt;   );   const linkElement = screen.queryByText(/Add image/);   expect(linkElement).toBeInTheDocument(); });</pre>
<b>Login Page</b>	<pre>import React from "react"; import { render, screen } from "@testing-library/react"; import LoginConnected from "../molecules/Login.jsx"; import { BrowserRouter } from "react-router-dom"; import { Provider } from "react-redux"; import store from "../redux/store";  test("renders Login Page", () =&gt; {   render(     &lt;Provider store={store}&gt;       &lt;React.StrictMode&gt;         &lt;BrowserRouter&gt;           &lt;LoginConnected /&gt;         &lt;/BrowserRouter&gt;       &lt;/React.StrictMode&gt;     &lt;/Provider&gt;   );   const linkElement = screen.queryByText(/Login/);   expect(linkElement).toBeInTheDocument(); });</pre>
<b>Header</b>	<pre>import React from "react"; import { render, screen } from "@testing-library/react"; import Header from "../molecules/Header.jsx"; import { BrowserRouter } from "react-router-dom"; import { Provider } from "react-redux"; import store from "../redux/store";  test("renders Header Component", () =&gt; {   render(     &lt;Provider store={store}&gt;       &lt;React.StrictMode&gt;         &lt;BrowserRouter&gt;           &lt;Header /&gt;         &lt;/BrowserRouter&gt;       &lt;/React.StrictMode&gt;     &lt;/Provider&gt;   );   const linkElement = screen.queryByText(/Etsy/);   console.log("hdhjbshhj",linkElement)   expect(linkElement).toBeInTheDocument(); });</pre>

## Product Tile

```
import React from "react";
import { render, screen } from "@testing-library/react";
import { BrowserRouter } from "react-router-dom";
import { Provider } from "react-redux";
import store from "../redux/store";
import ProductItem from "../atom/ProductItem.jsx";

const prodconst = {"category": "https://i5.walmartimages.com/asr/93bb1cce-4af9-48f0-8c0e-b8aabe78b58b.0d1293e7fe185df77d6d4bc3cd193", "id": 15, "idshop": 15, "name": "Mainstays 8-Piece Black Bed", "photo": "https://i5.walmartimages.com/asr/93bb1cce-4af9-48f0-8c0e-b8aabe78b58b.0d1293e7fe185df77d6d4bc3cd193", "productPrice": 160, "purchaseQuantity": 2, };

test("renders Product Tile", () => {
  render(
    <Provider store={store}>
      <React.StrictMode>
        <BrowserRouter>
          <ProductItem productItem={prodconst} favlist={[]} />
        </BrowserRouter>
      </React.StrictMode>
    </Provider>
  );
  const linkElement = screen.queryByText("Mainstays 8-Piece Black Bed");
  expect(linkElement).toBeInTheDocument();
});
```

## Cart Item

```
import React from "react";
import { render, screen } from "@testing-library/react";
import { BrowserRouter } from "react-router-dom";
import { Provider } from "react-redux";
import store from "../redux/store";
import CartItem from "../atom/CartItem.jsx";

const prodconst = {"category": "https://i5.walmartimages.com/asr/93bb1cce-4af9-48f0-8c0e-b8aabe78b58b.0d1293e7fe185df77d6d4bc3cd193", "id": 15, "idshop": 15, "name": "Mainstays 8-Piece Black Bed", "photo": "https://i5.walmartimages.com/asr/93bb1cce-4af9-48f0-8c0e-b8aabe78b58b.0d1293e7fe185df77d6d4bc3cd193", "productPrice": 160, "purchaseQuantity": 2, };

test("renders Cart Item", () => {
  render(
    <Provider store={store}>
      <React.StrictMode>
        <BrowserRouter>
          <CartItem product={prodconst} />
        </BrowserRouter>
      </React.StrictMode>
    </Provider>
  );
  const linkElement = screen.queryByText("Quantity:");
  expect(linkElement).toBeInTheDocument();
});
```

## Git Commit history

The screenshot shows a GitHub repository page for 'Snichat97/EtsyPrivateAWS'. The 'Code' tab is selected, displaying the 'main' branch. The commit history is as follows:

- Commits on Apr 24, 2022:
  - Update Profile Display. (1bf1565) - Snichat97 committed 2 hours ago
- Commits on Apr 23, 2022:
  - User profile fix. (4fe81db) - Snichat97 committed 4 hours ago
  - API fixing . (3aa4d8b) - Snichat97 committed 5 hours ago
  - IP fixes. (7223bf5) - Snichat97 committed 7 hours ago
  - Shifting to AWS large instance IP change. (ec5c211) - Snichat97 committed 8 hours ago
  - Port fixes for kafka. (2e899e9) - Snichat97 committed 9 hours ago
  - Changing client/server of kafka to AWS public IP. (af87715) - Snichat97 committed 9 hours ago
  - Fix of import library (6a42bc2) - Snichat97 committed 9 hours ago

## Performance:

Performance testing was done using JMeter. Visible across all loads (100,200,300,400,500 users) the connection pooling of 500 connections outperformed the single connection. It is significantly notable in heavier loads.

With 100 user test case

- Connection Pooling | 0.3 ms average throughput
- Single connection | 3 ms average throughput

With 500 user test case

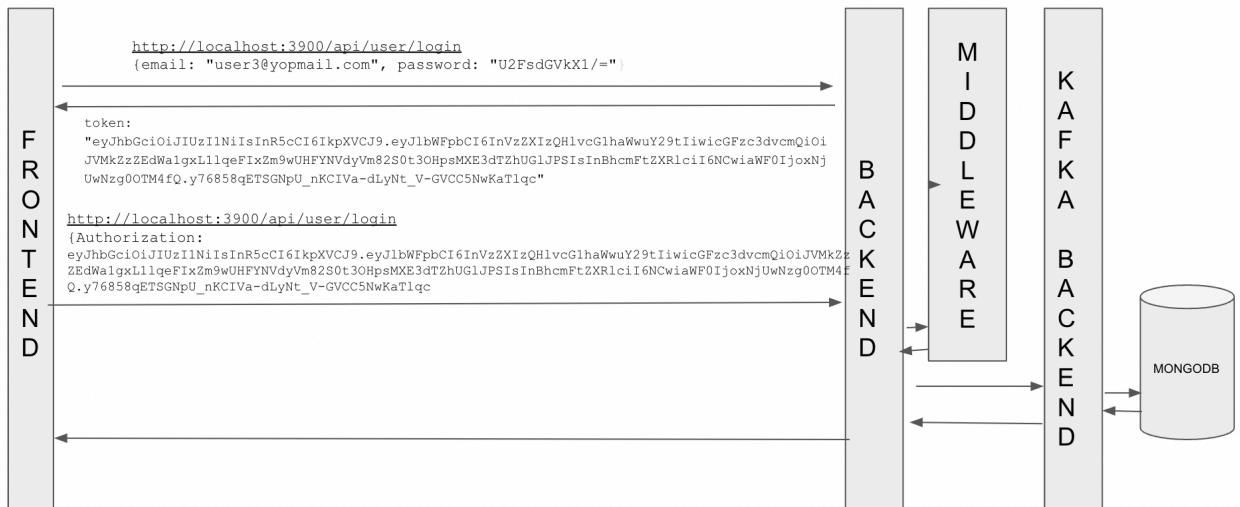
- Connection Pooling | 1 ms average throughput
- Single connection | 613 ms average throughput

## Questions

Please find the questions for Lab2 which you have to answer in your lab report

1. Compare the passport authentication process with the authentication process used in Lab1.

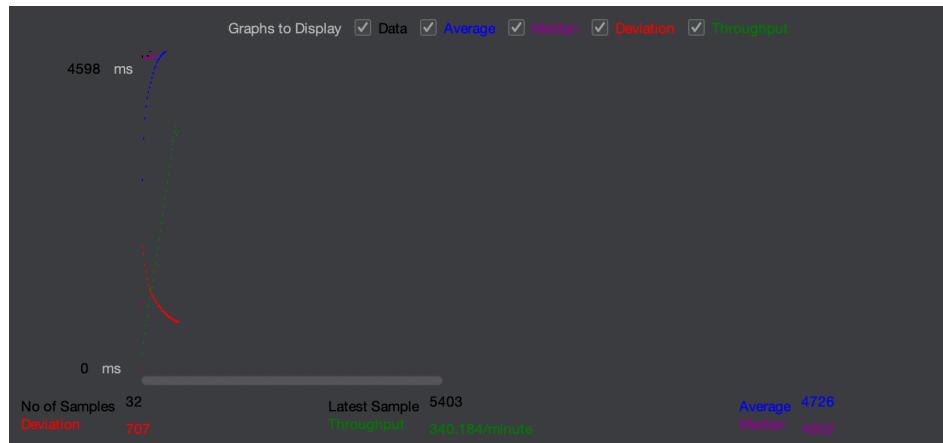
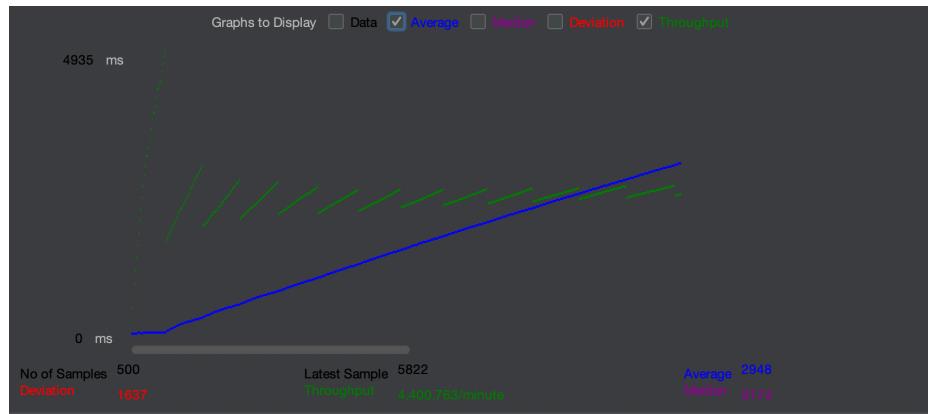
In the current flow of the application when the user logs in or signs up for the first time an authentication token is returned, which is carried forward in all the subsequent API calls. Before the API is executed the call is sent to the middleware. If the token is valid the API is executed else an error is sent to the frontend.



2. Compare performance with and without Kafka. Explain in detail the reason for the difference in performance.

Kafka significantly enhances the performance of the Application. This is visible in various parameters :

	Kafka	Backend Directly with MongoDB
Throughput	4400.763/ min	340.184/min
Messages Dropped	0	Multiple queries resulted in unavailable server
Average Response Time	2948 ms	4726 ms



3. Given an option to implement MySQL and MongoDB both in your application, specify which part of data of the application you will store in MongoDB and MySQL respectively

I would use MySQL for user, shop, and product tables as they all are following strict relational schema and for various functionalities. Also for these tables (favorite products: user - product, shop products: shop - product), join functionality is leveraged.

For Orders, I would use MongoDB as it has a flexible schema that may or may not have Gift Boolean or Gift Message. It also has some old data maintained for products due to the dynamic schema MongoDB is more ideal for this usecase.