ZING

A MEAN STACK ECOMMERCE STORE

DONE BY :

AAKAR MUTHA : BFDL

INTRODUCTION

Due to the pandemic, many of us have started to buy things online rather than visiting a physical store whenever possible. Using an online ecommerce platform has made buying the products with better deals a reality. With an online store, a seller can widen their audience which in turn results in more revenue. The seller can manage the products that he has on sale, give special discounts on products, sell products from different sellers on their platform and many more. Due to these advantages, having and maintaining an online store has become an essential task for any merchant.

In this Case Study I aimed at making a fully functional Ecommerce website. A customer can add products to cart and purchase them while a seller can see the stats in real time, add products, update products as well as any normal user add products to cart and purchase them.

TECH STACK

1) MONGODB - 4.4.0

MongoDB is an open source NoSQL database management program. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information. MongoDB supports various forms of data. It is one of the many nonrelational database technologies that arose in the mid-2000s under the NoSQL banner - normally, for use in big data applications and other processing jobs involving data that doesn't fit well in a rigid relational model. Instead of using tables and rows as in relational databases, the MongoDB architecture is made up of collections and documents.

Organizations can use MongoDB for its ad-hoc queries, indexing, load balancing, aggregation, server-side JavaScript execution and other features.

2) EXPRESS JS - 4.17.2

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework −

- Allows to set up middlewares to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

Since it is a framework of Node.js , most of the code is already written for programmers to work with. You can build a single page, multi-page, or hybrid web applications using Express.js. Express.js is lightweight and helps to organize web applications on the server-side into a more organized MVC architecture.

3) ANGULAR - 13.2.1

Angular is a development platform, built on TypeScript. As a platform, Angular includes:

- A component-based framework for building scalable web applications
- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
- A suite of developer tools to help you develop, build, test, and update your code

With Angular, you're taking advantage of a platform that can scale from single-developer projects to enterprise-level applications. Angular is designed to make updating as straightforward as possible, so take advantage of the latest developments with a minimum of effort. Best of all, the Angular ecosystem consists of a diverse group of over 1.7 million developers, library authors, and content creators.

4) NODE JS 16.13.2 (LTS)

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.
This is in contrast to today's more common concurrency model, in which OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node.js are free from worries of dead-locking the process, since there are no locks. Almost no function in Node.js directly performs I/O, so the process never blocks except when the I/O is performed using synchronous methods of Node.js standard library. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

FEATURES

USER DASHBOARD

1) View all products from all sellers at a glance.
2) Add Products to cart once the user is logged in.
3) View product details.
4) View Cart.
5) Delete items from cart.
6) Make Payment.

SELLER DASHBOARD

1) View statistics like Total Orders, Total orders shipped, Total products shipped.
2) Add New Products.
3) View All the products sold by them.
4) Edit the product details like title, price and description
5) All the functionalities of the user.

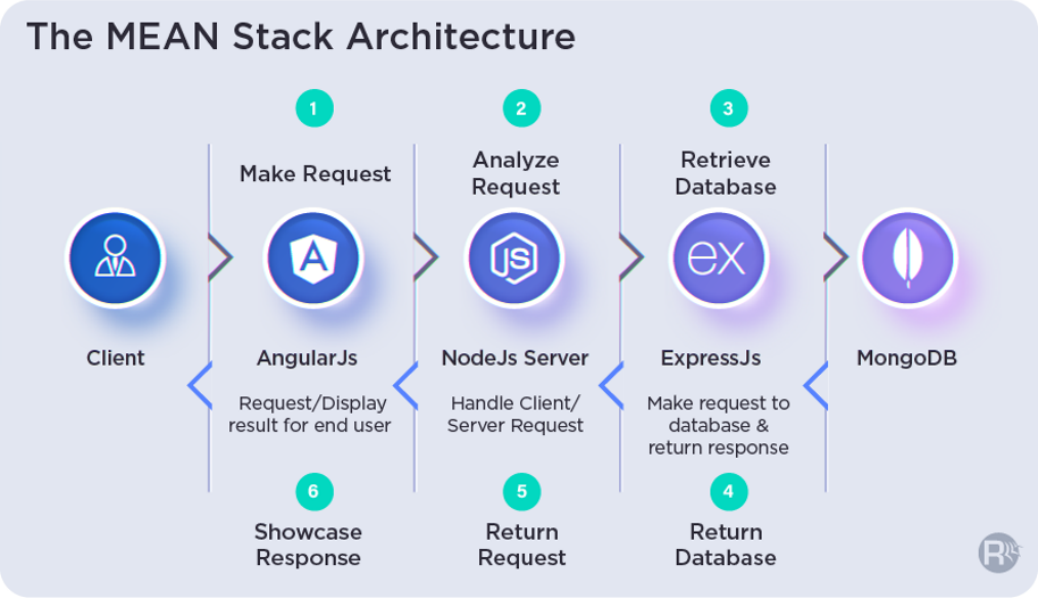
ANGULAR CONCEPTS USED

- Each Dashboard is in a different module
- Lazy loading for the different modules
- Reactive Forms to handle form data with validation
- Template Forms.
- Role Guards.
- Routing

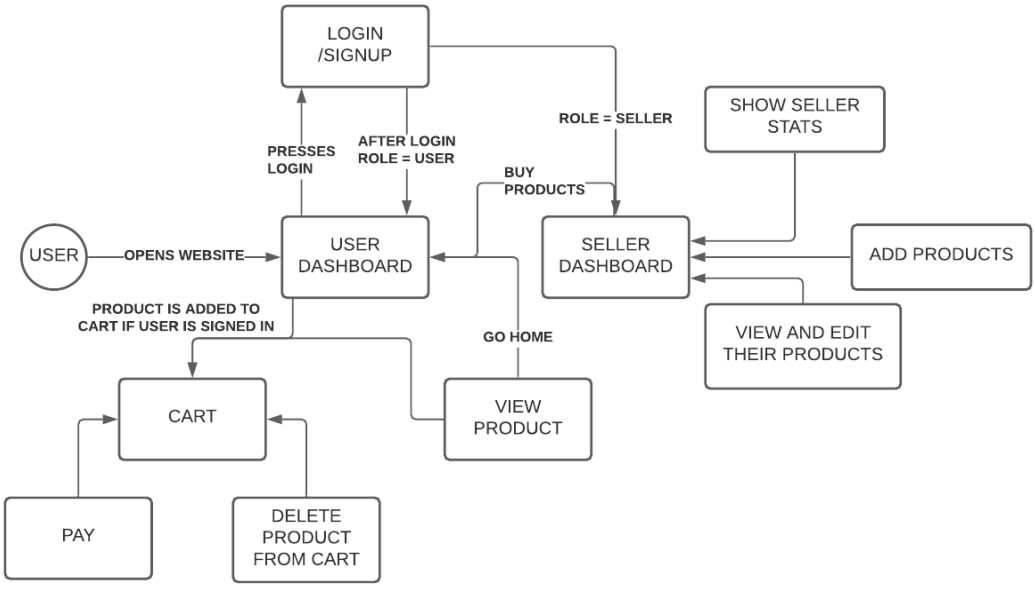
LIBRARIES USED

- Ng-bootstrap - 11.0.0
- Ng-fontawesome - 5.15.4
- Mongoose - 6.2.0
- Multer - 1.4.4
- Express - 4.17.2
- Cors - 2.8.5
- Angular Material - 13.2.1

ARCHITECTURE DIAGRAM



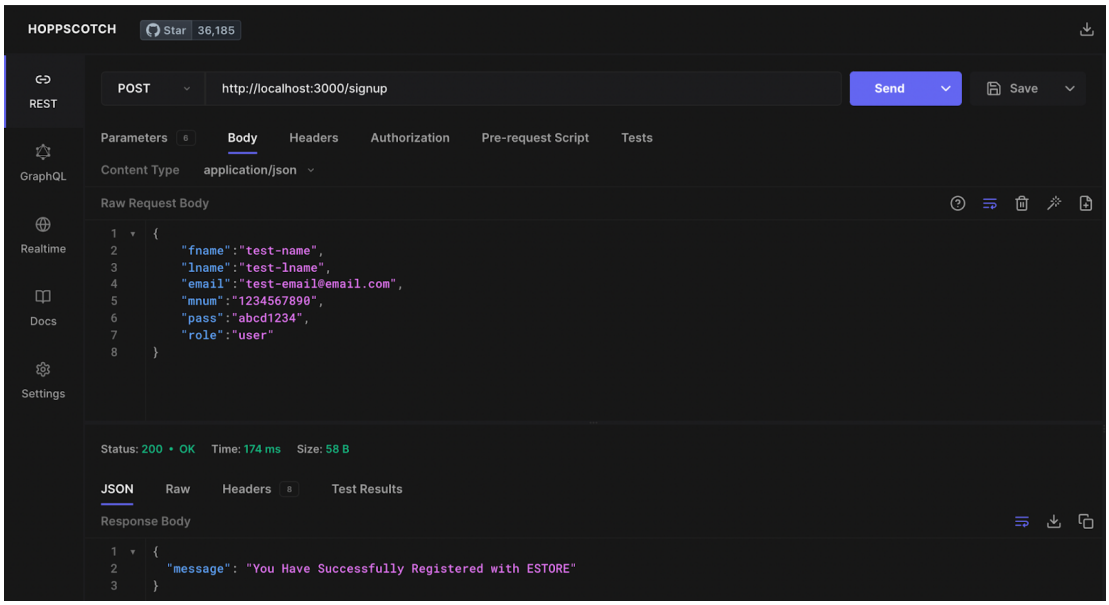
COMPONENT DIAGRAM



API TESTS

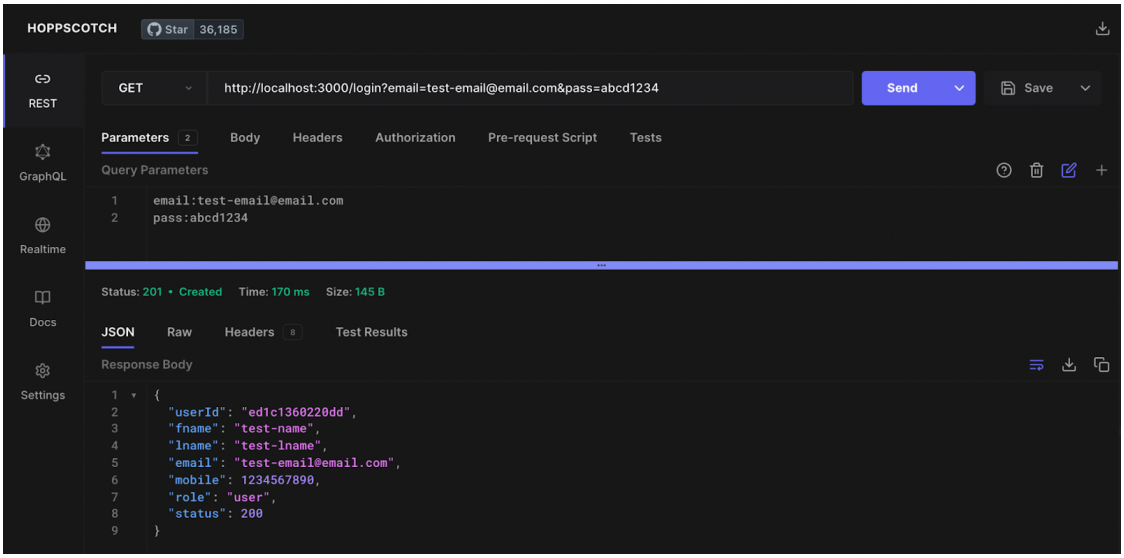
1) POST - /signup

Used to sign up a new user.



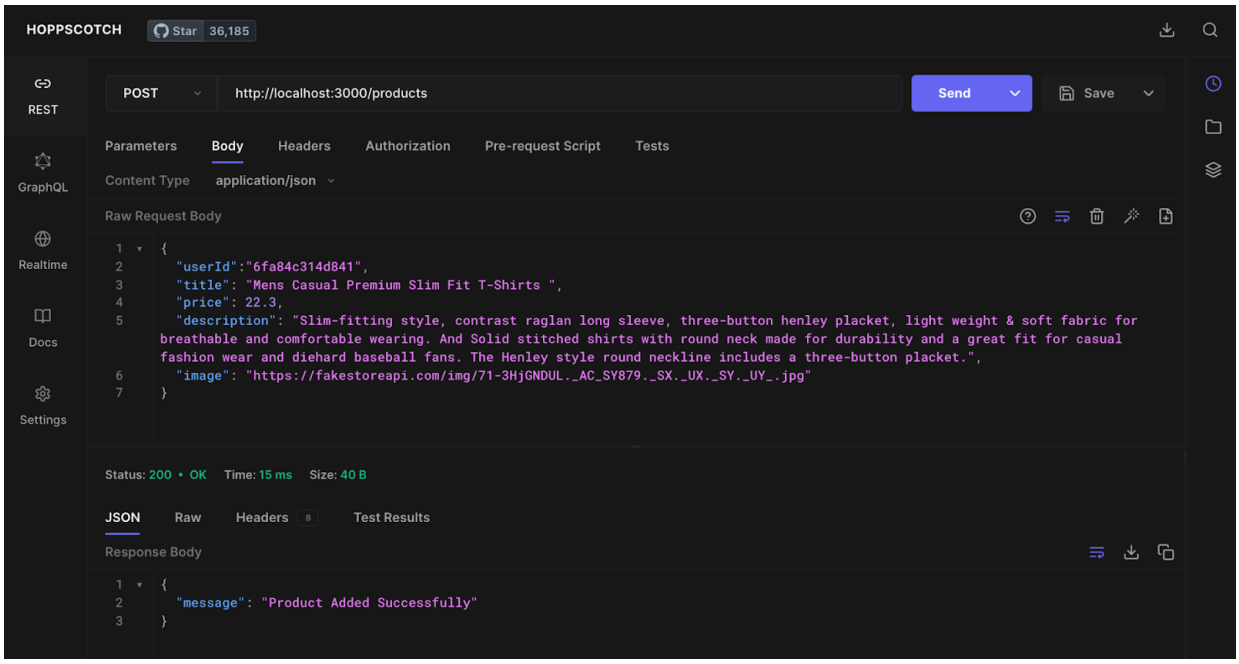
2) GET - /login

Used to login a user into the app.



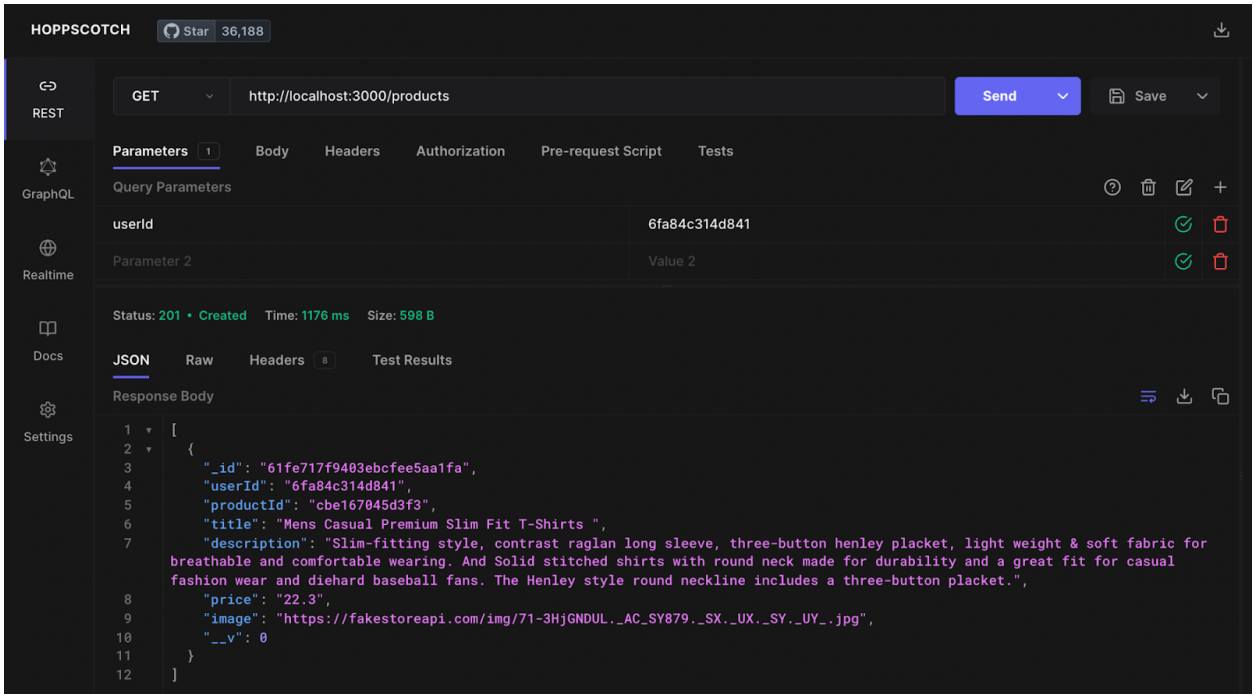
3) POST - /products

Used to add products to cart



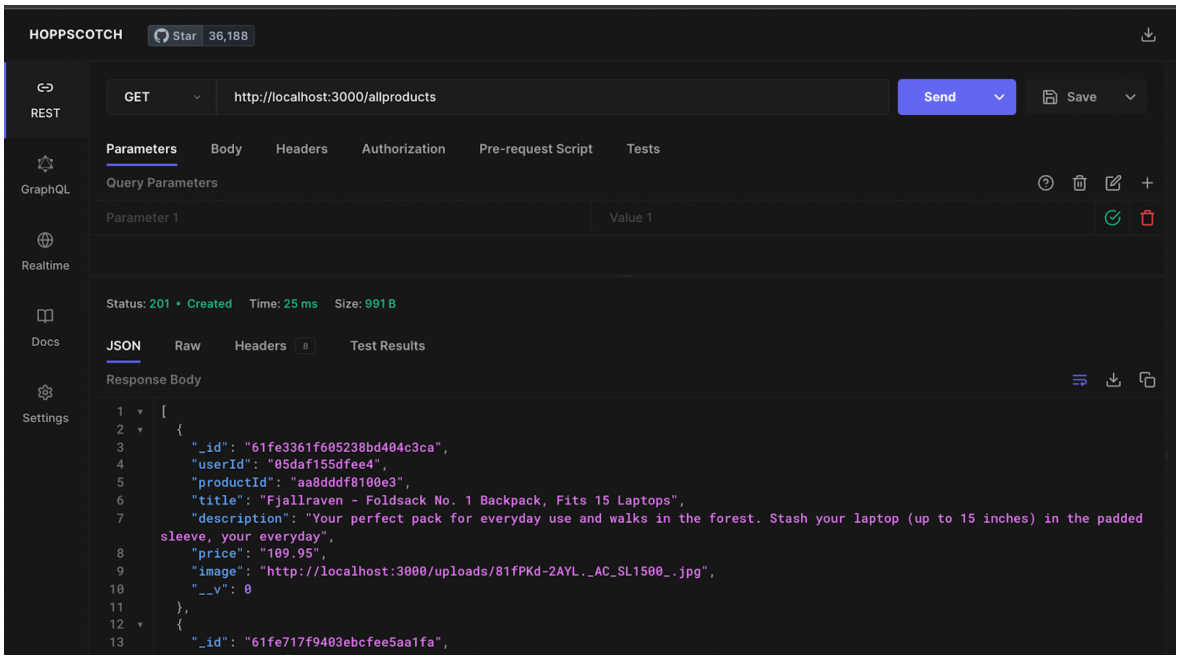
4) GET - /products

Used to get all products of a particular seller with userId.



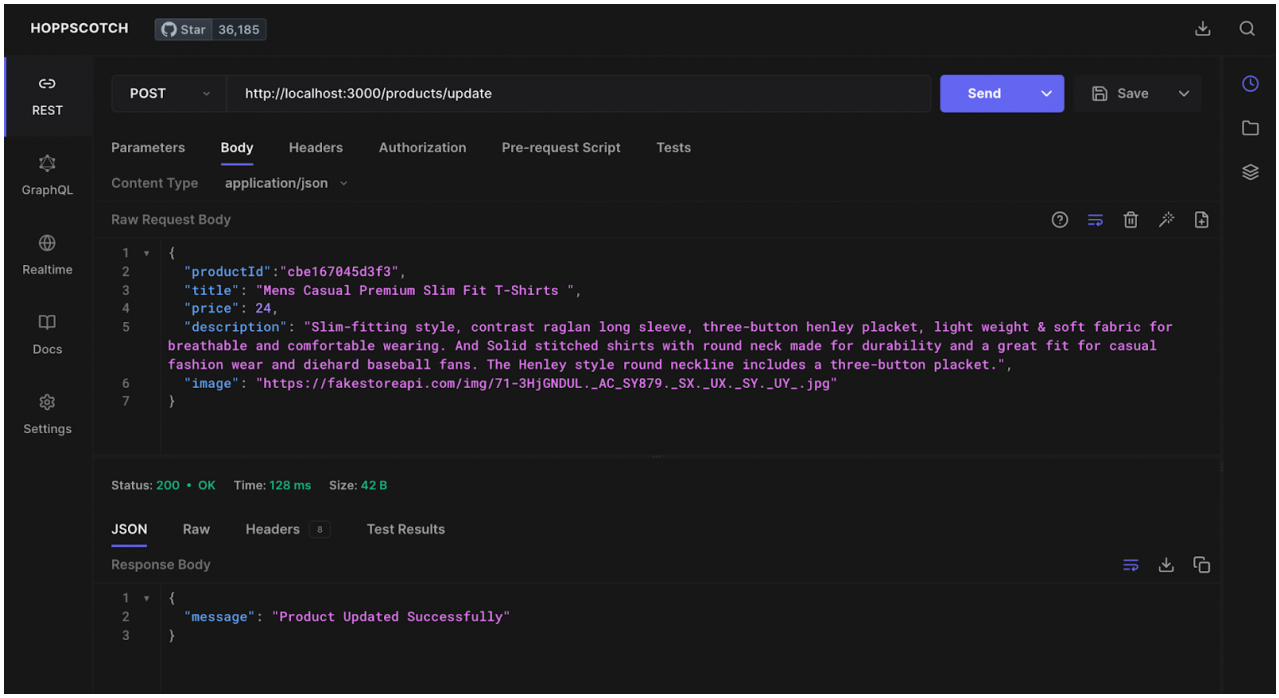
5) GET - /allproducts

Used to get all the products to show to the user.



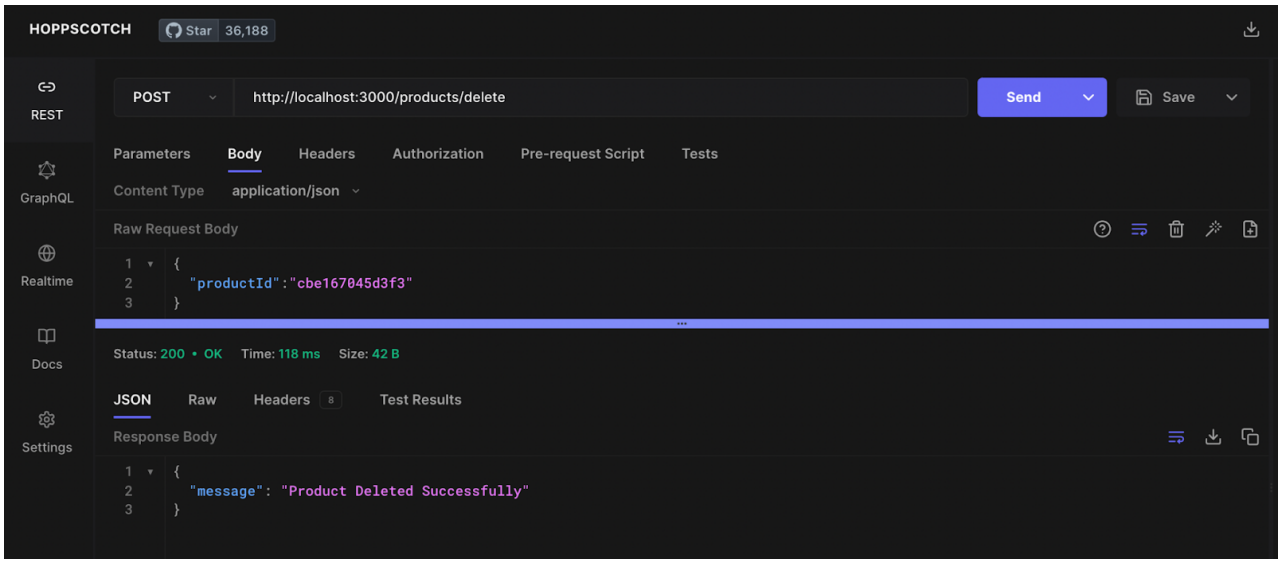
6) POST - /products/update

Used to update the product details by the seller.



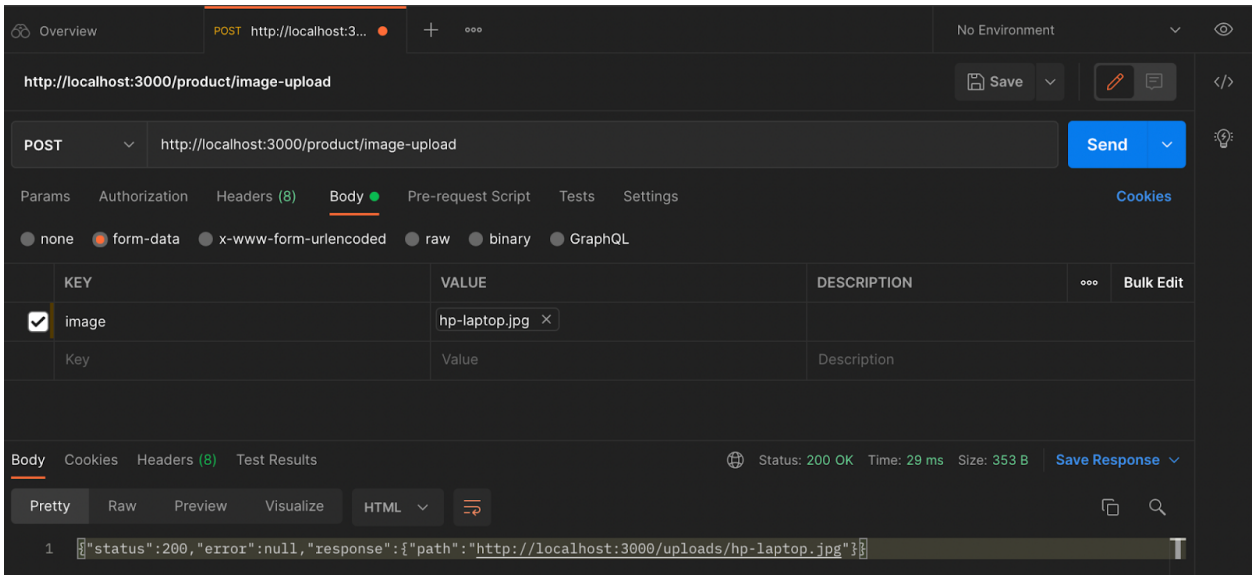
7) POST - /products/delete

Used to delete a product by the seller



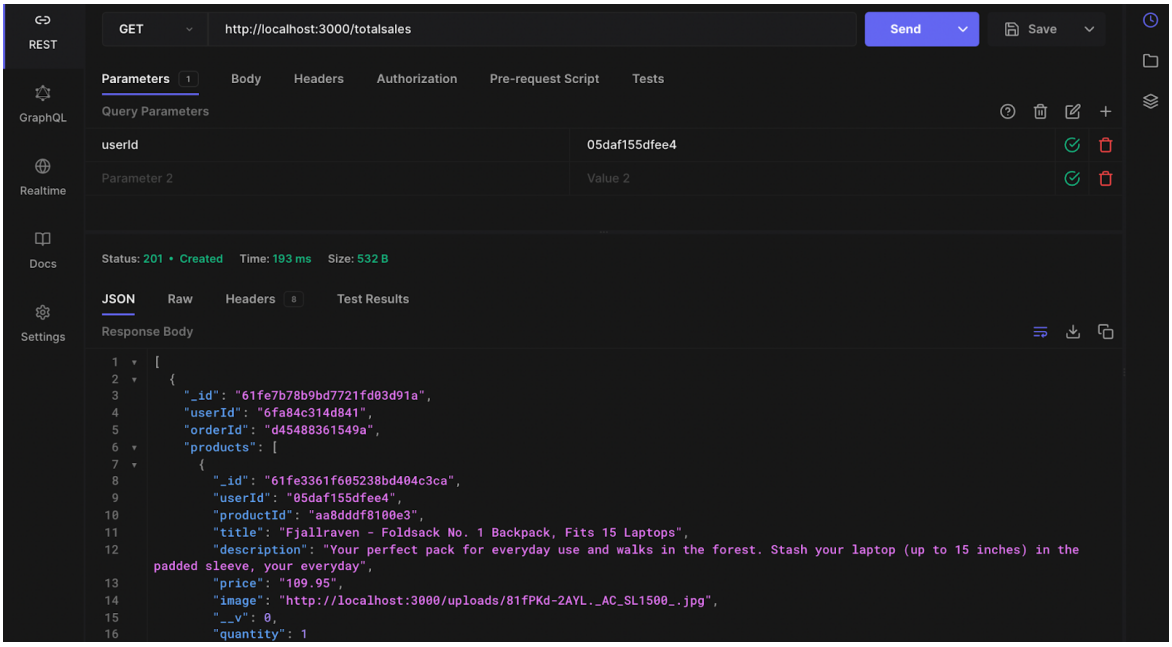
8) POST - /products/image-upload

Used to add the image of the product by the seller.



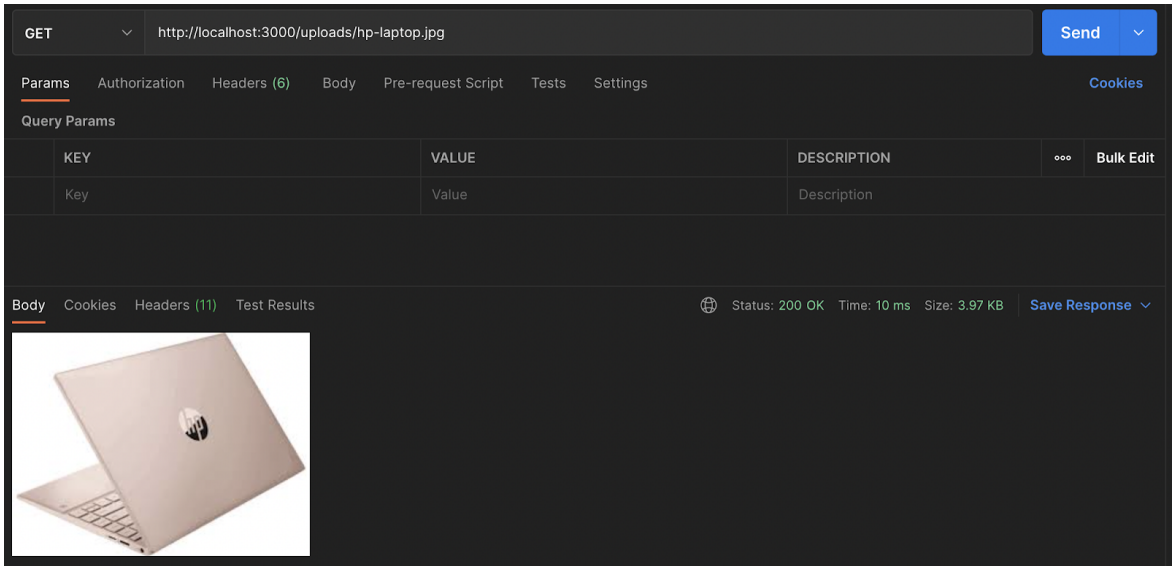
9) GET - /totalsales

Returns the products that are of the seller to display in the seller dashboard.



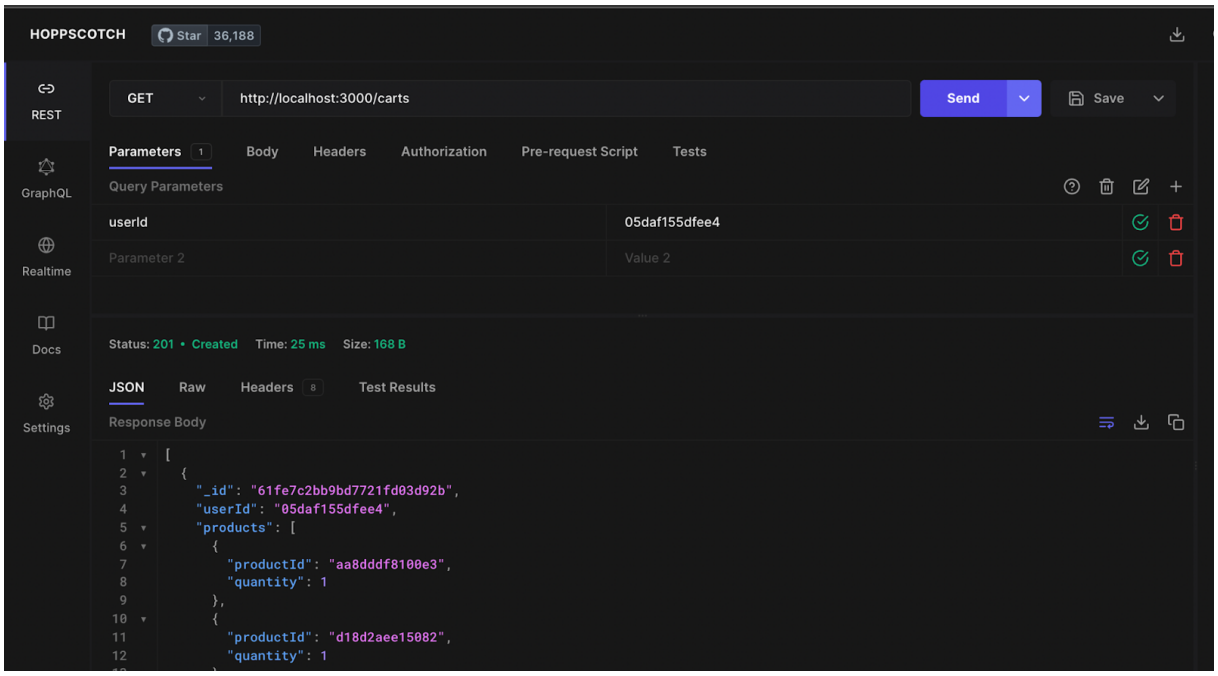
10) GET - /uploads

Used to get the image of a product to display in the website



11) GET - /carts

Used to get the cart of a particular user



12) POST - /carts/delete

Used to delete item from cart

13) POST - /carts

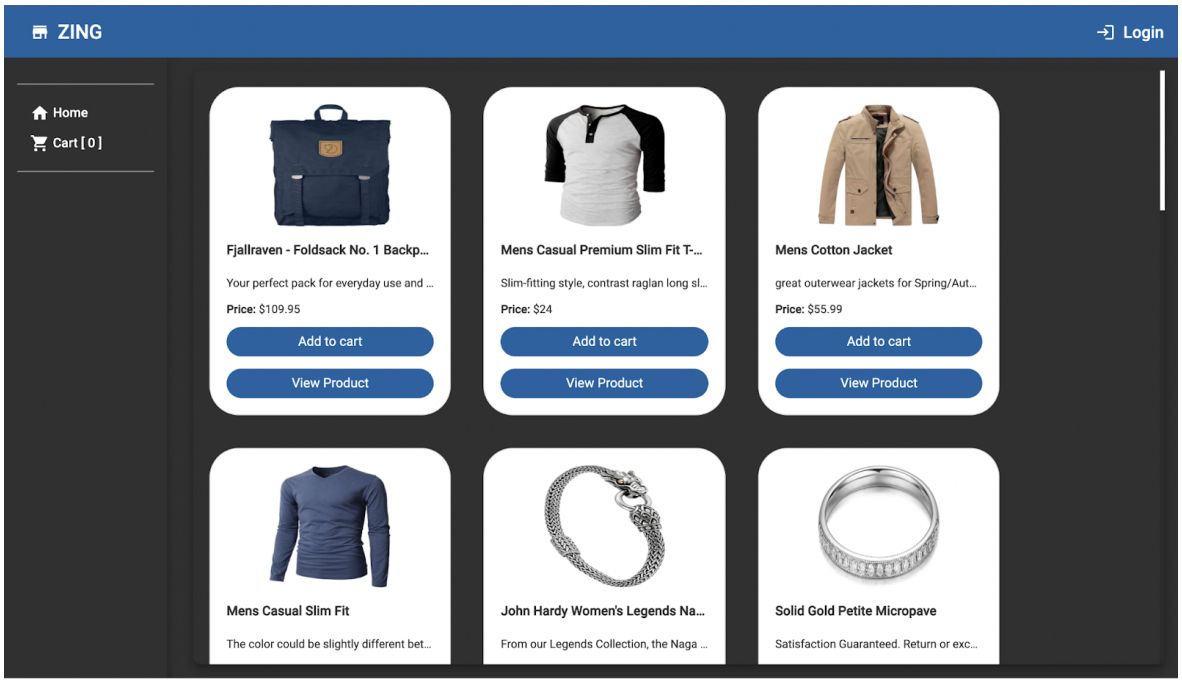
Used to add items to cart for a particular user

14) POST - /carts/pay

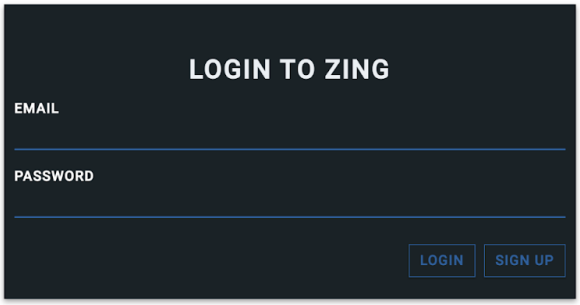
Used when the user makes payment and the payment is successful.

APPLICATION SCREENS

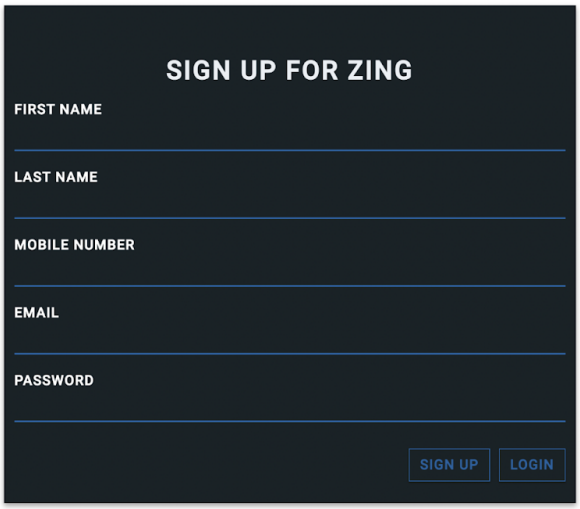
Home Screen



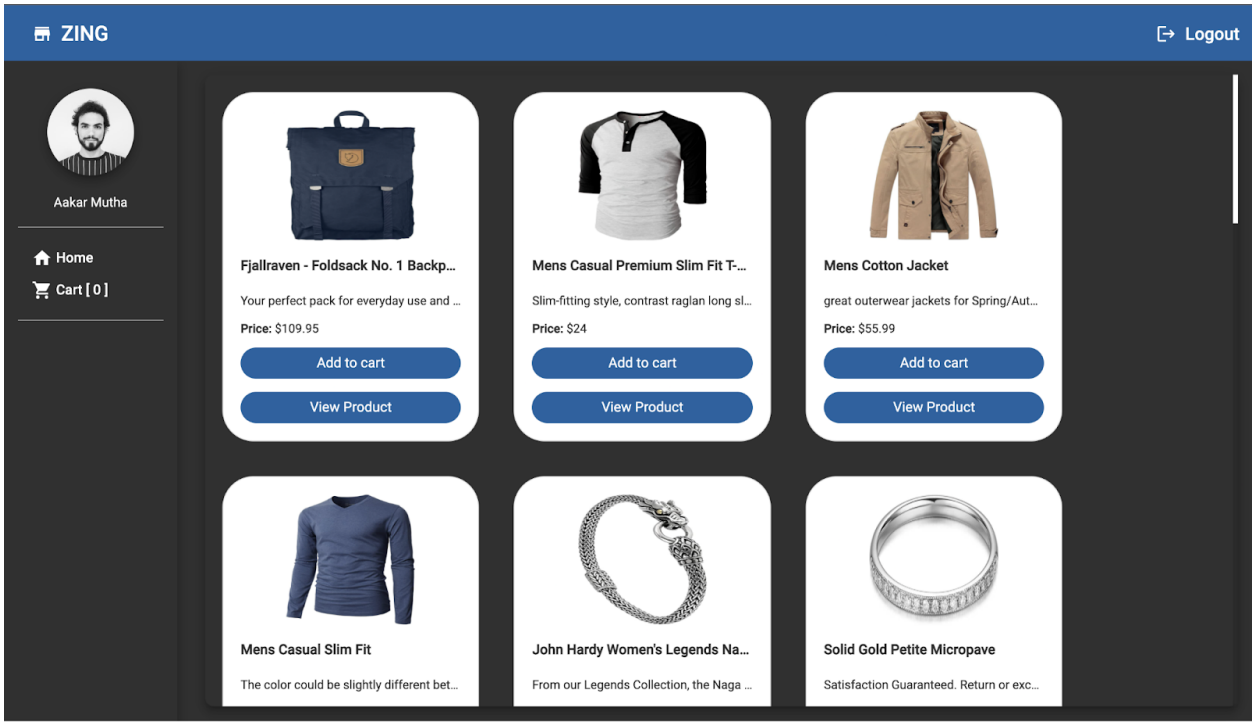
Login Screen



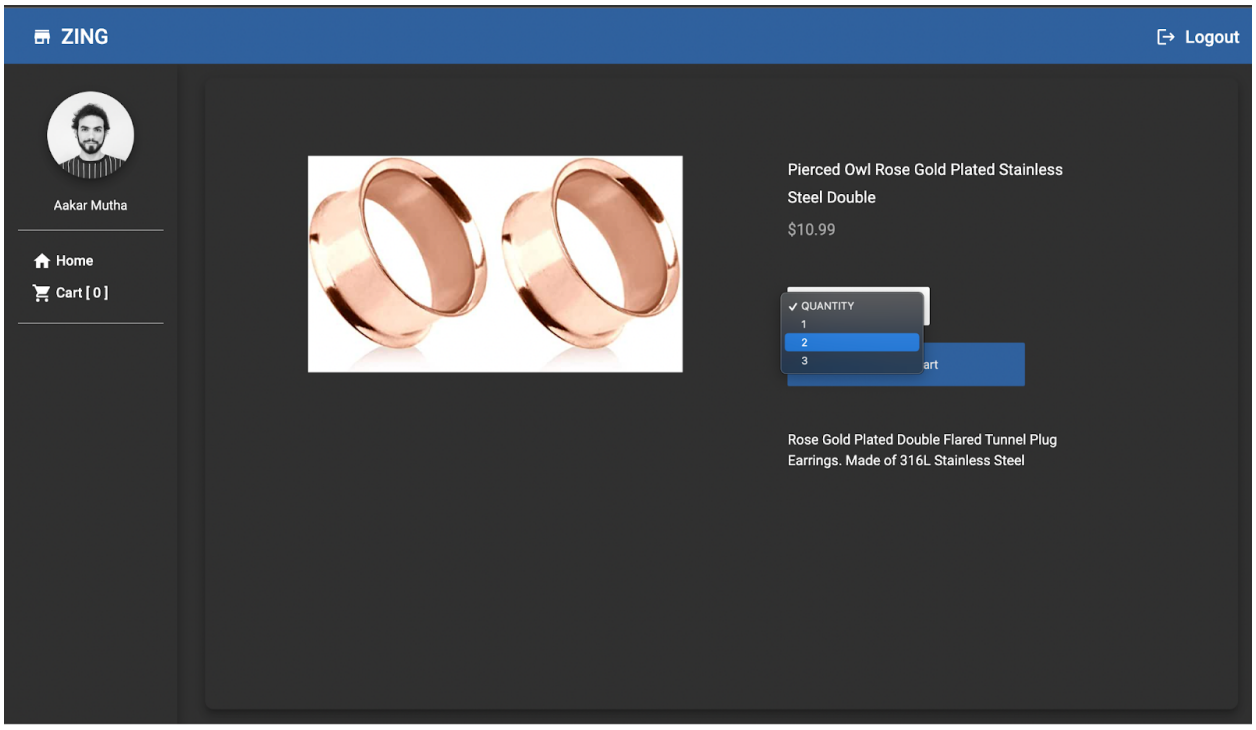
Signup Screen



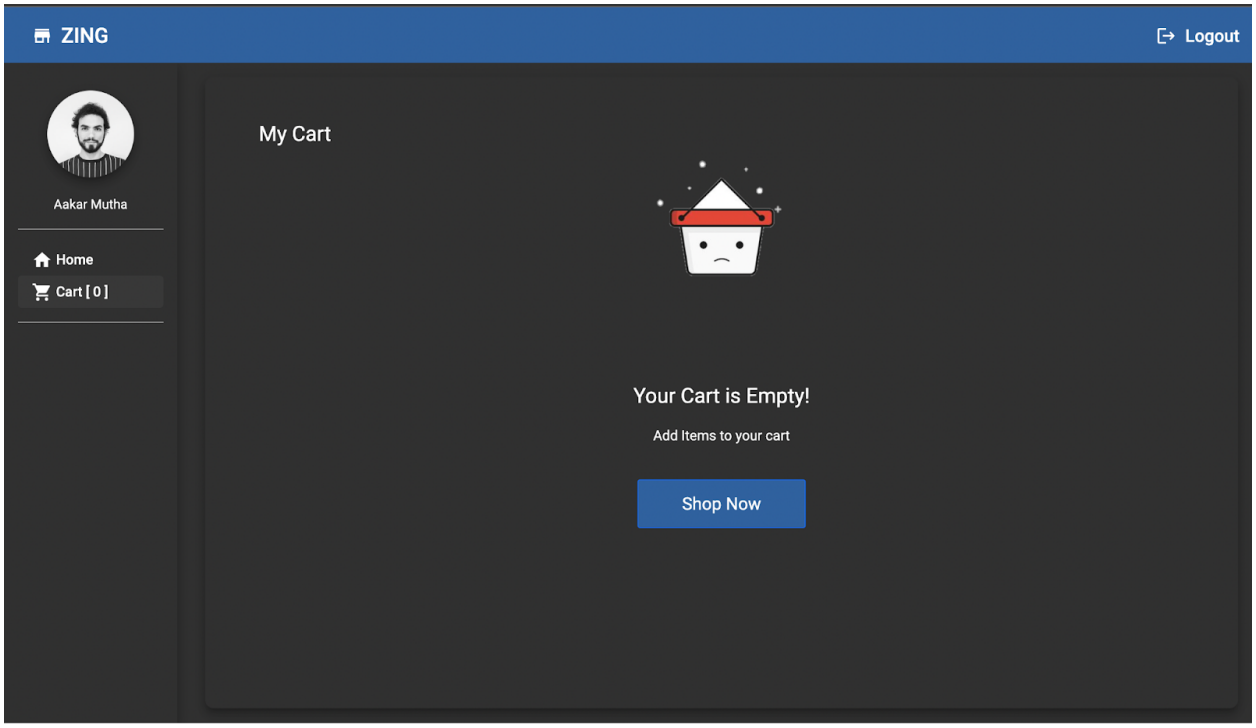
When the User is Logged In



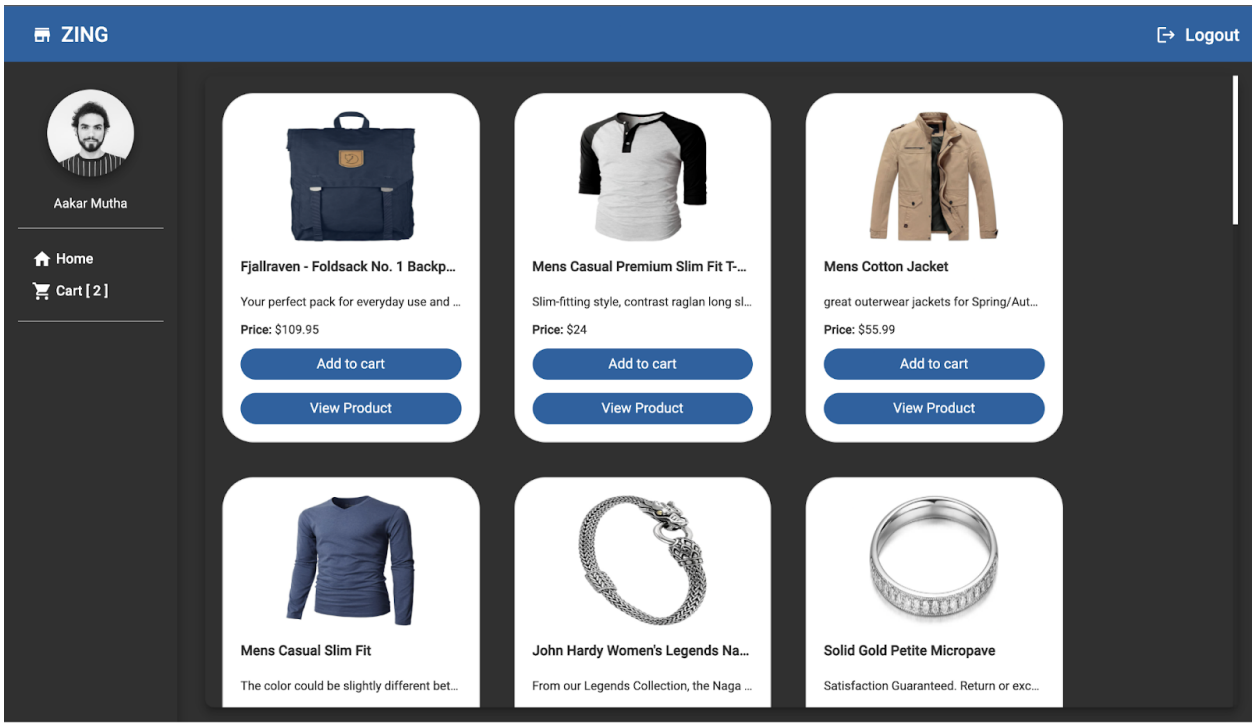
View Product Details



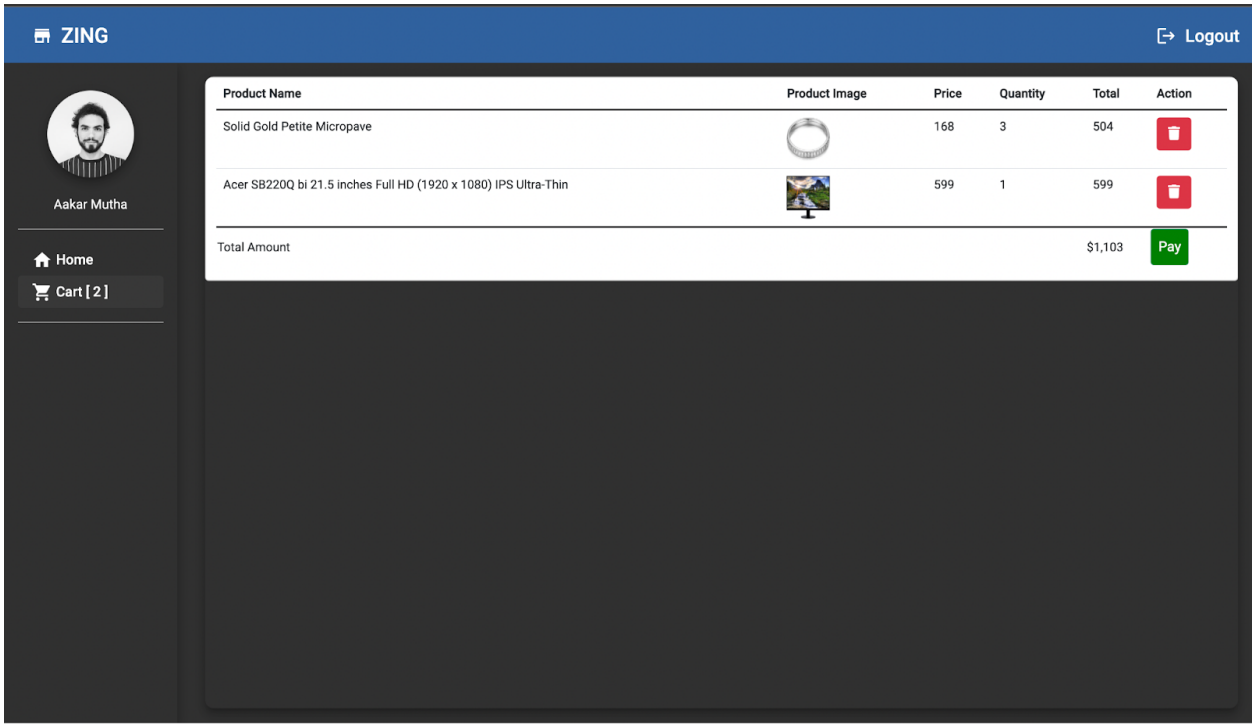
Empty Cart



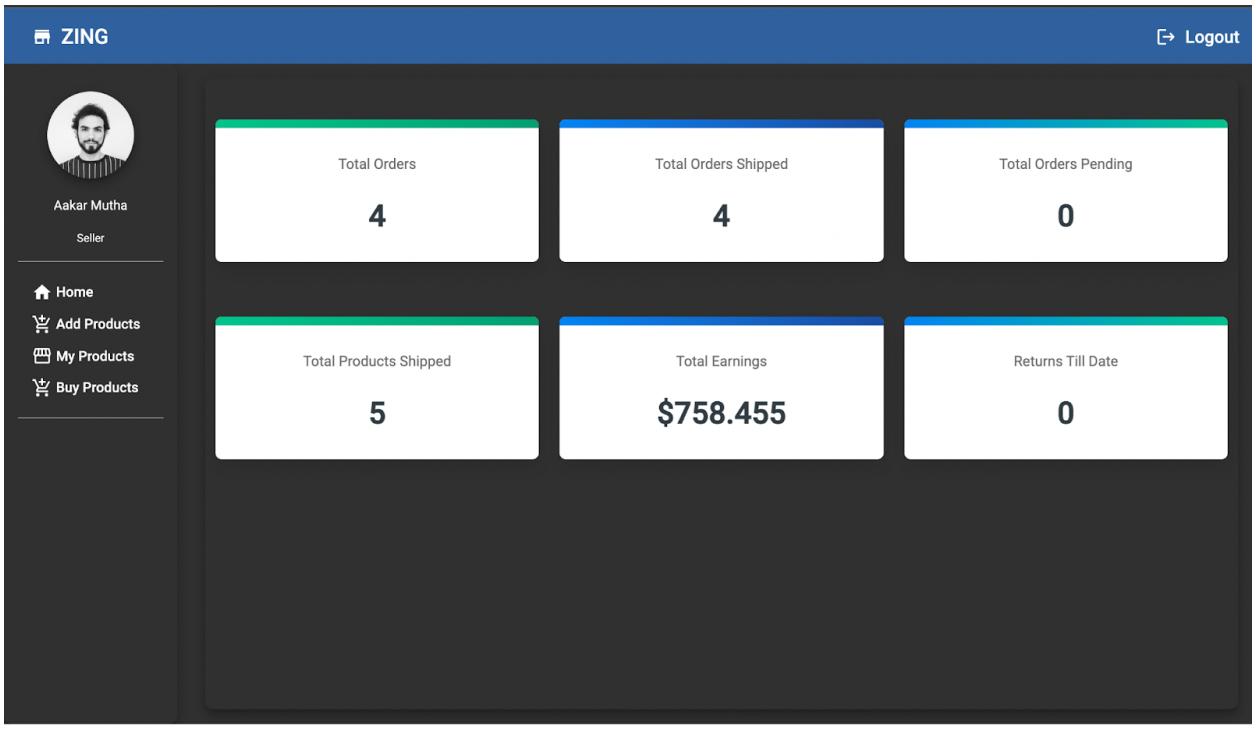
Home Page When there are items in the Cart



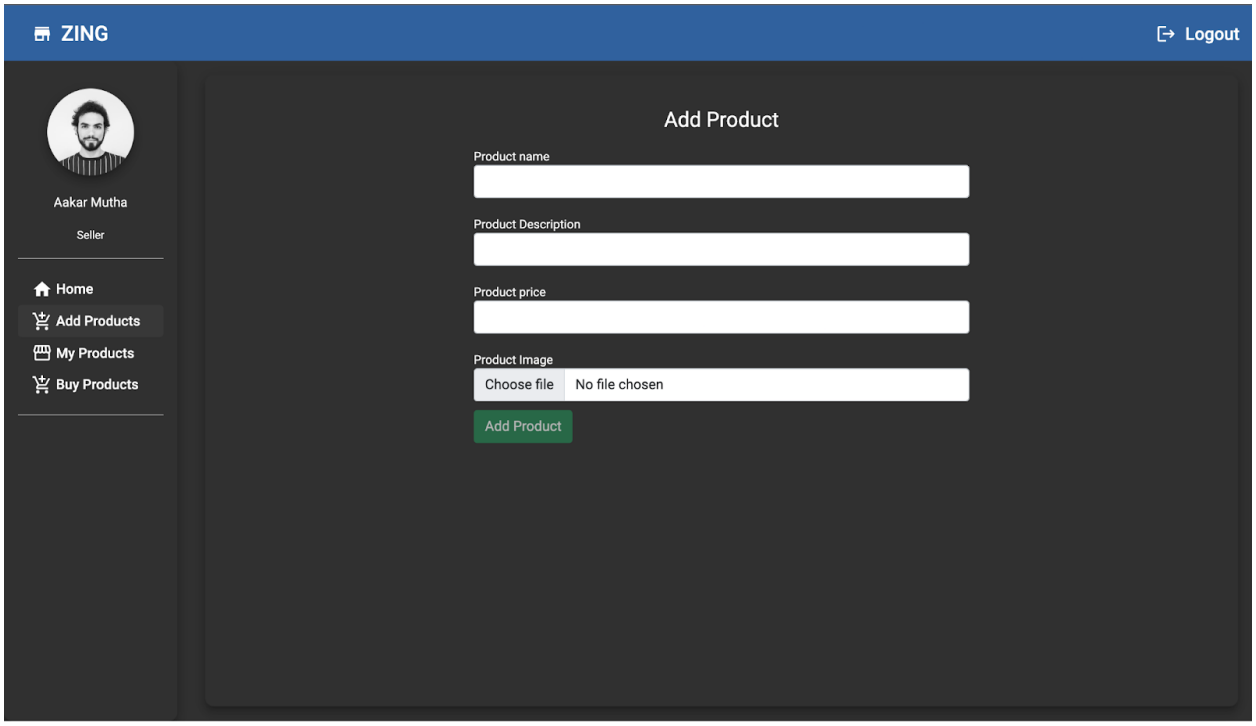
Cart when there are item in it



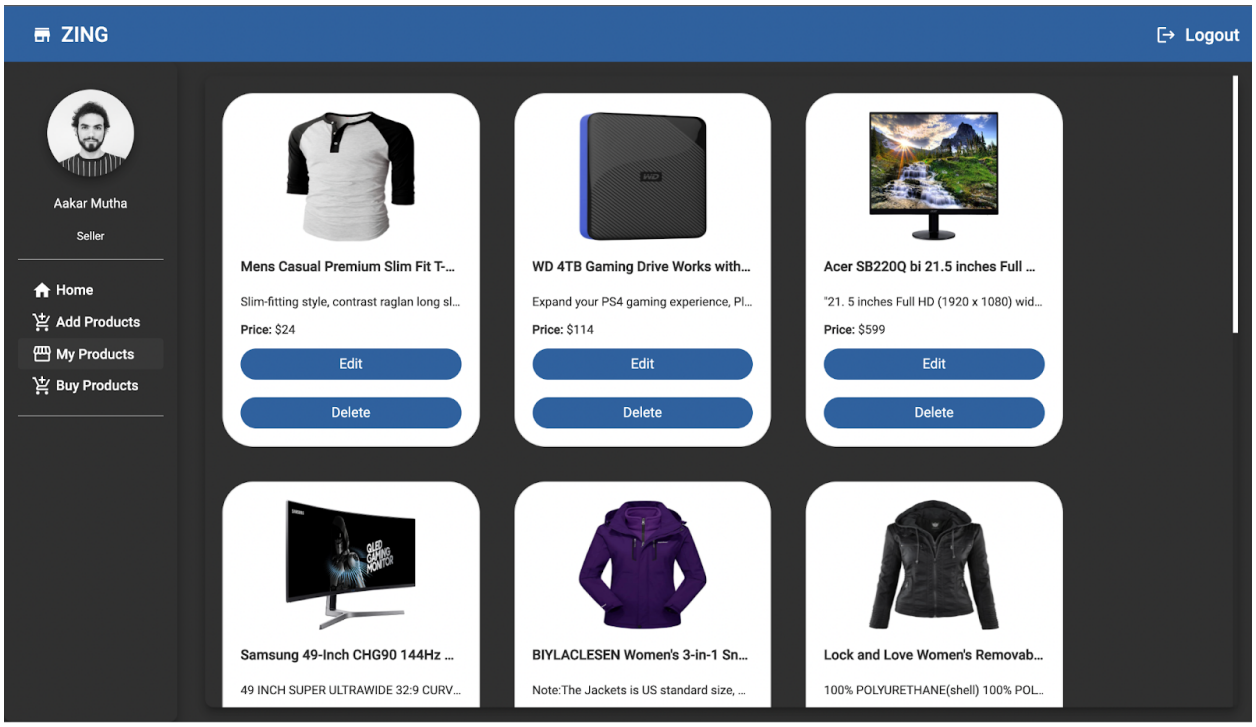
Seller Dashboard



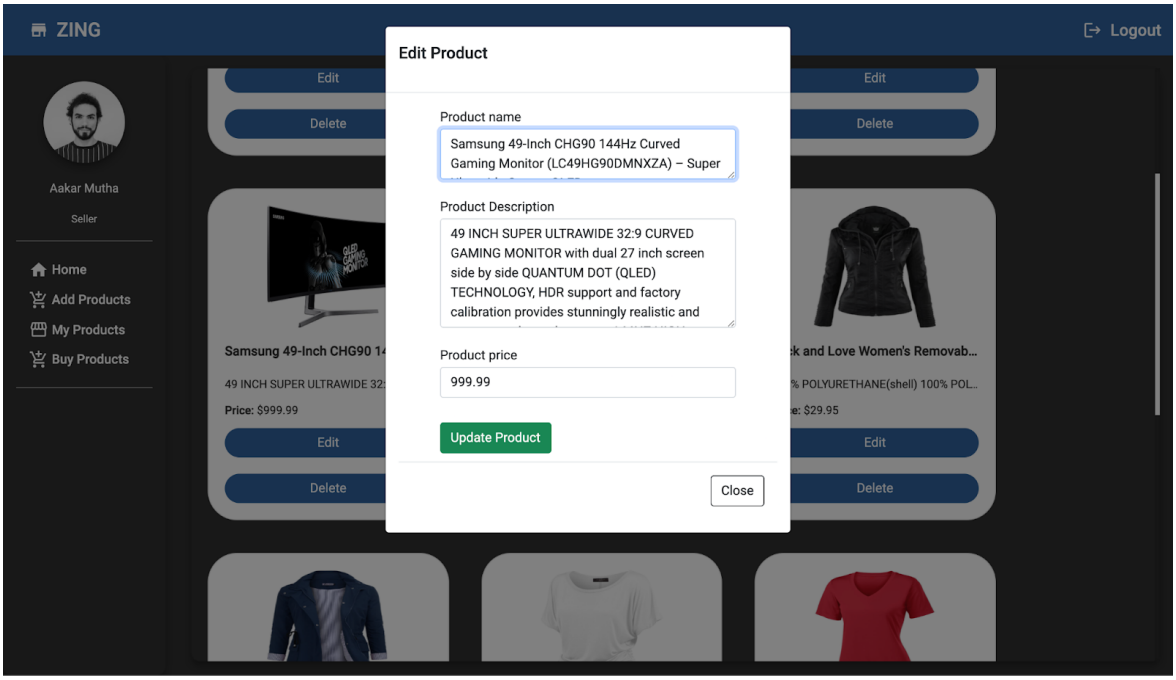
Add Products



My Products

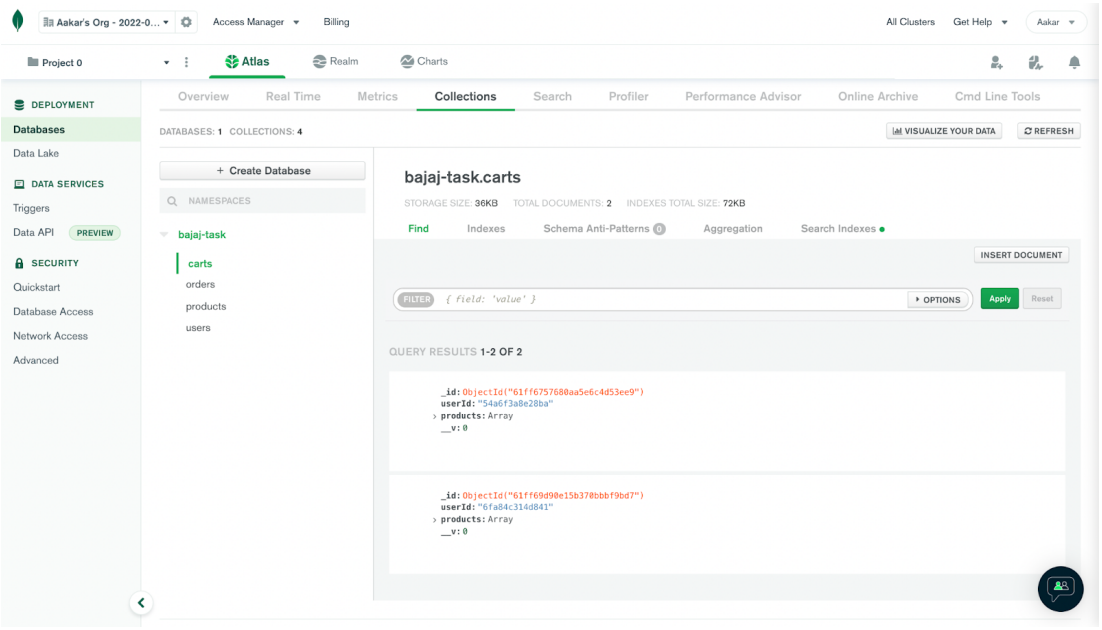


Update Product Modal

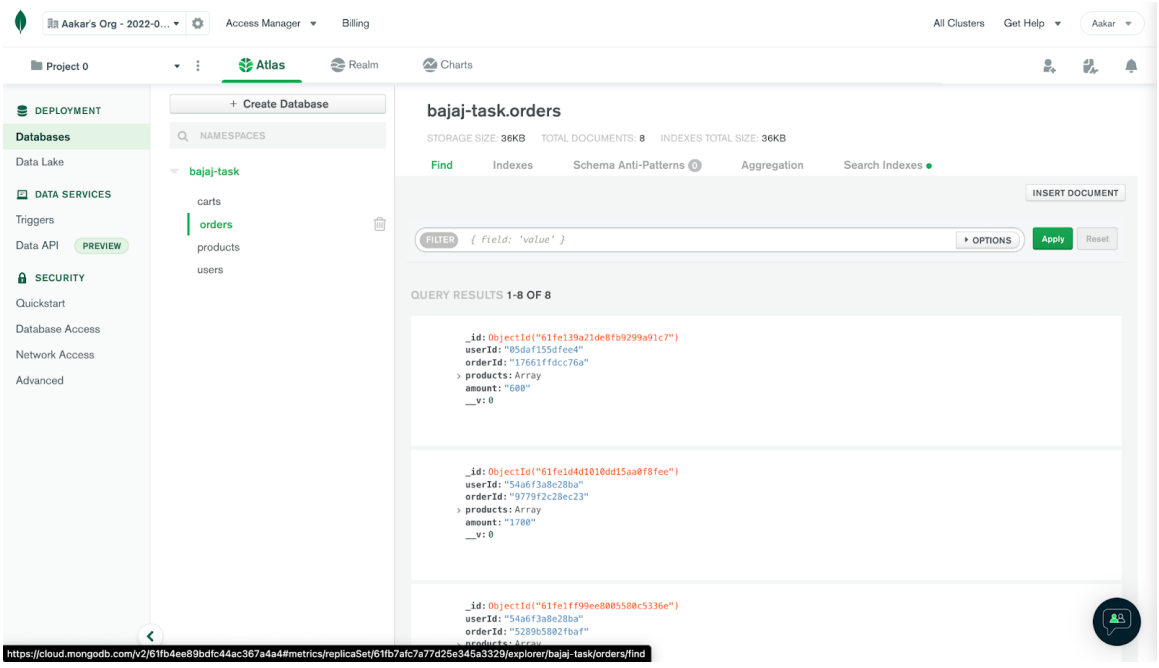


MONGODB ATLAS VIEW

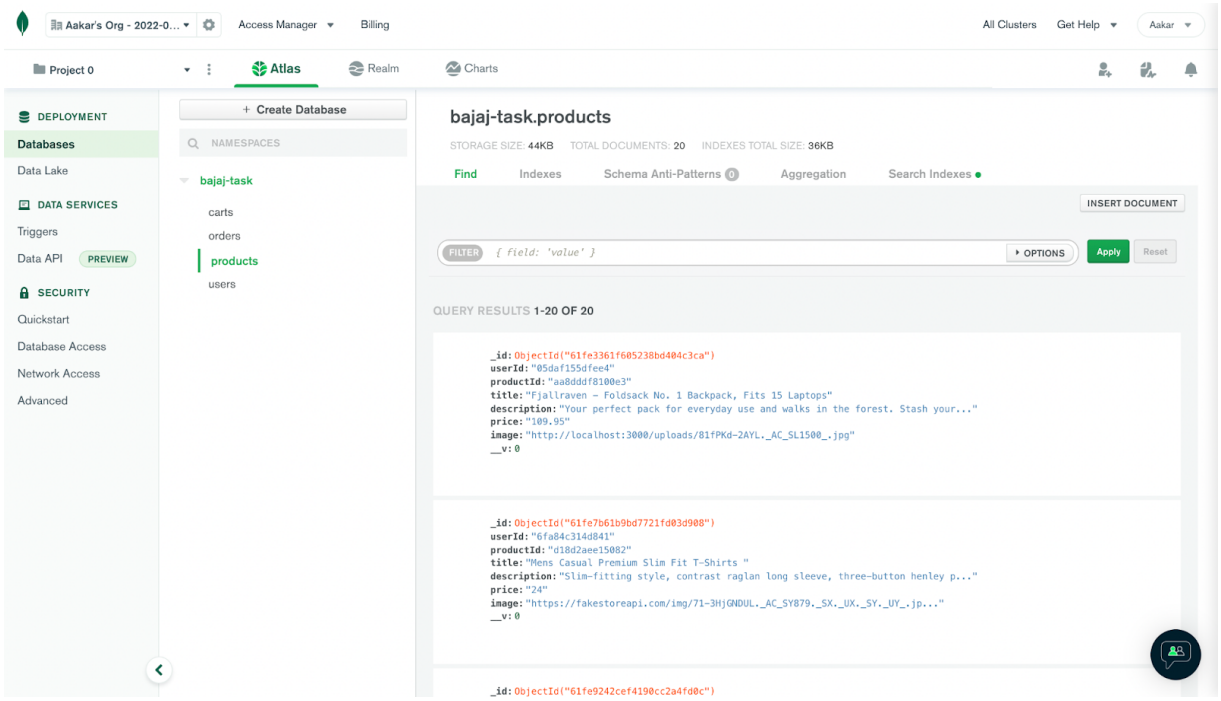
Carts Collection



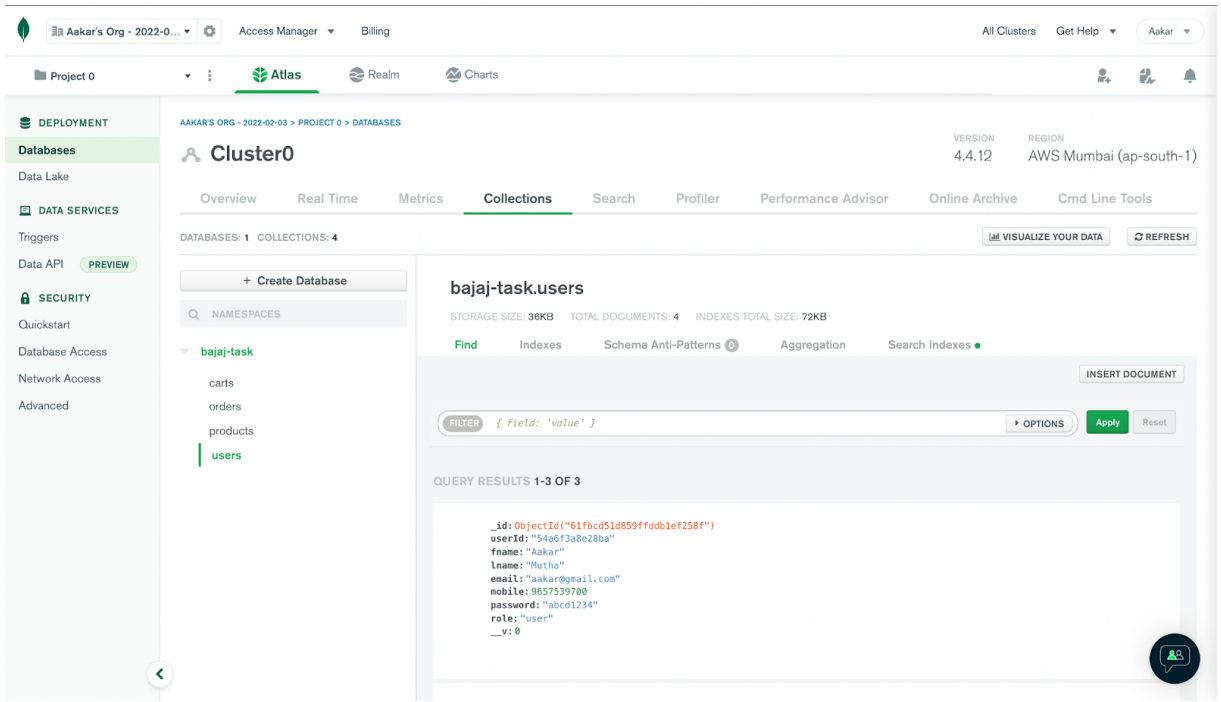
Orders Collection



Products Collection

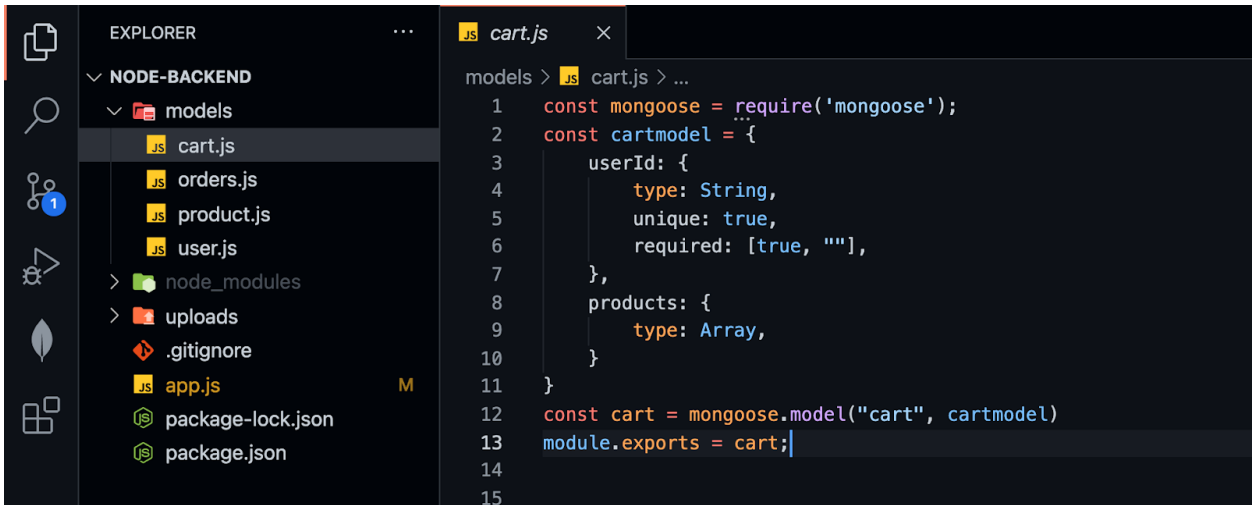


Users Collection

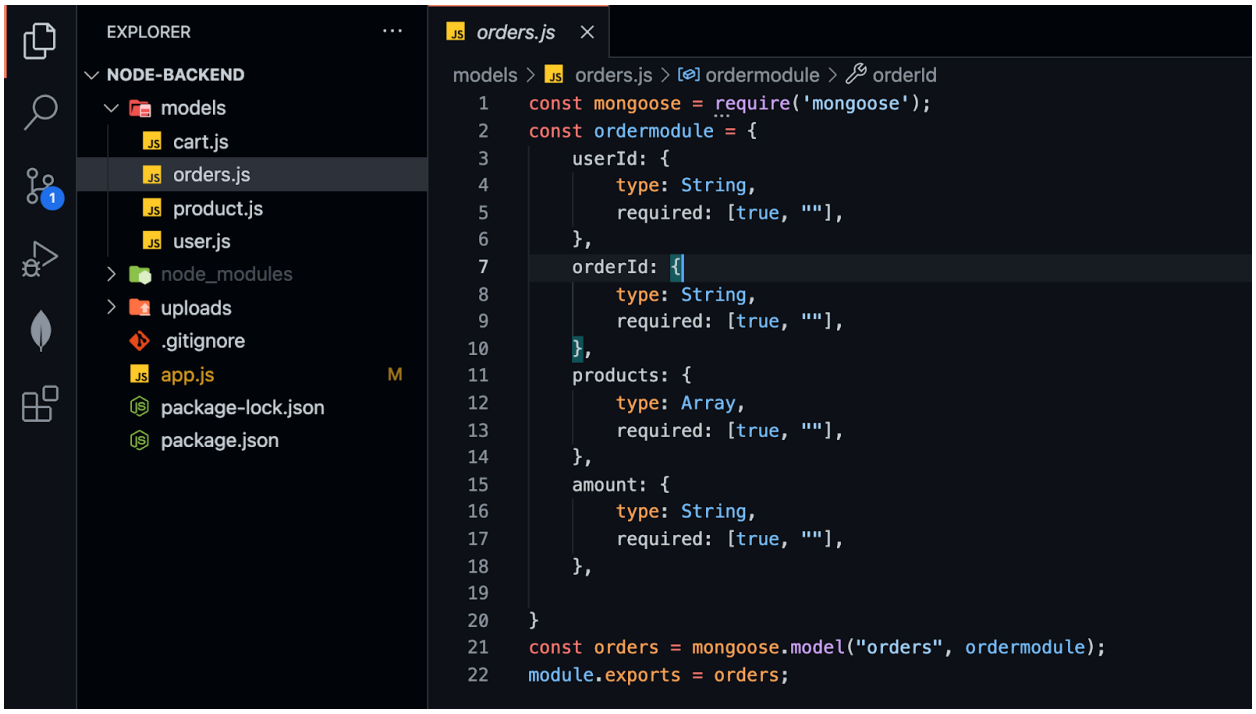


Models

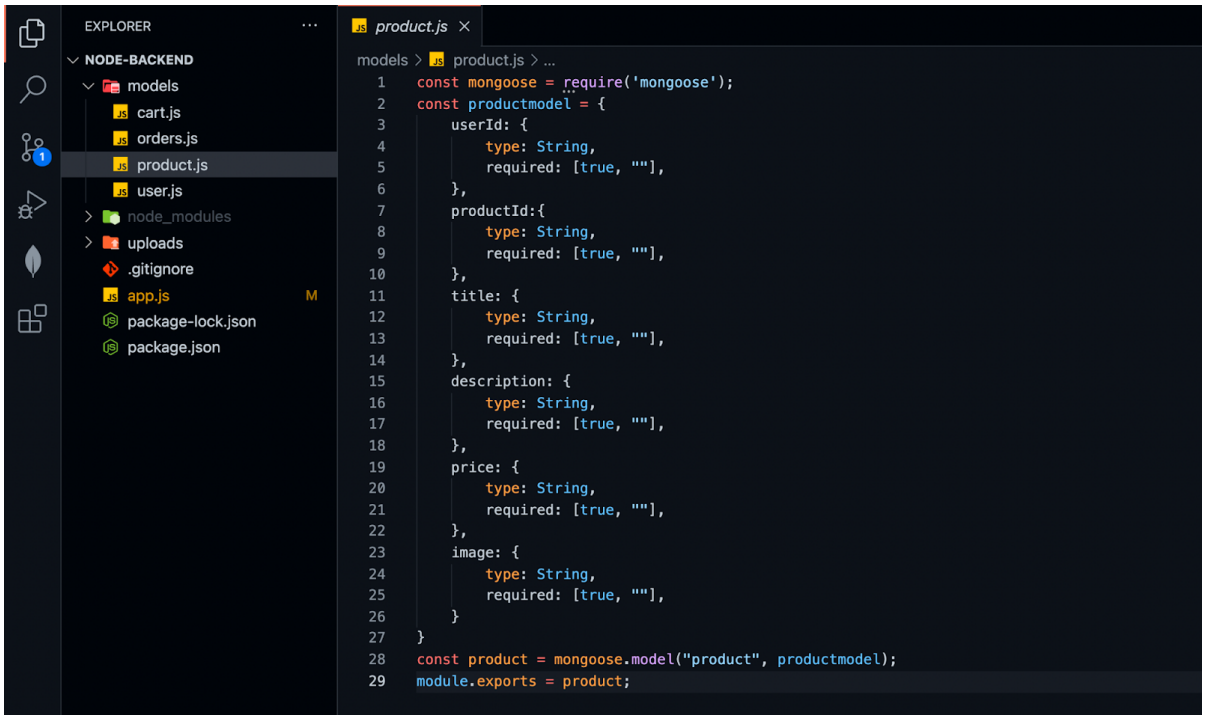
Cart Model



Orders Model



Products Model

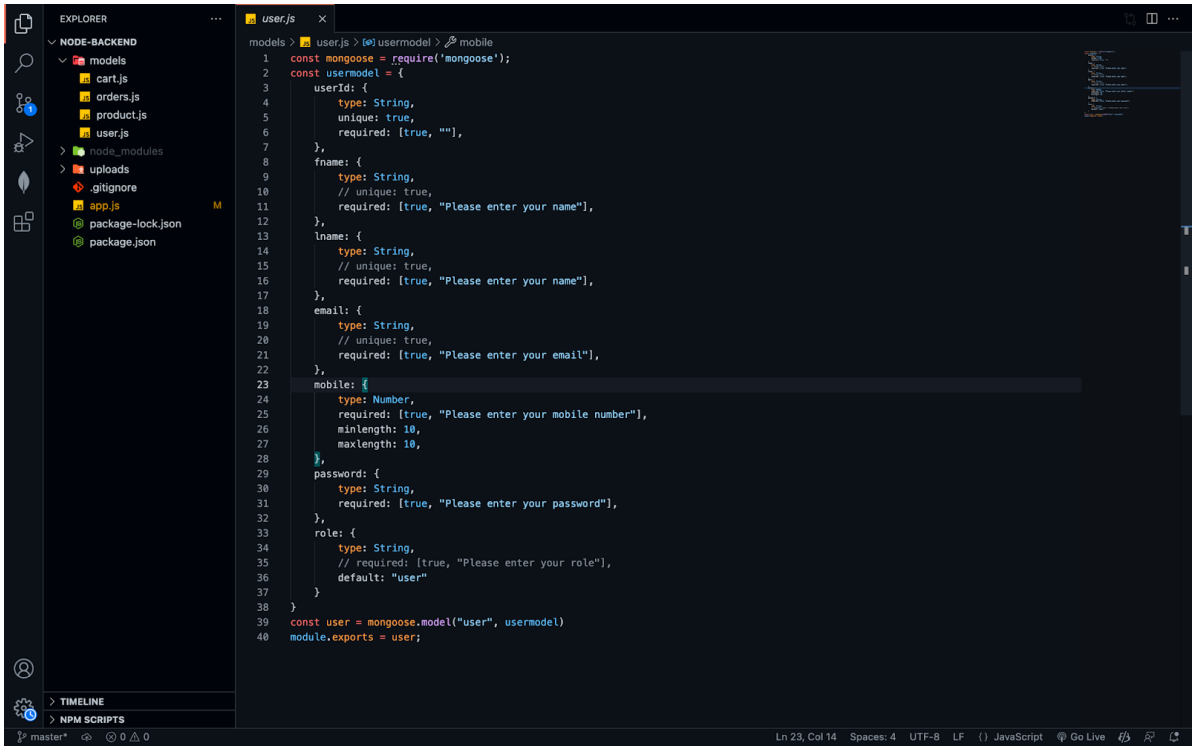


```
EXPLORER
NODE-BACKEND
models
cart.js
orders.js
product.js
user.js
node_modules
uploads
.gitignore
app.js M
package-lock.json
package.json
```

product.js

```
models > product.js > ...
1  const mongoose = require('mongoose');
2  const productmodel = {
3      userId: {
4          type: String,
5          required: [true, ""],
6      },
7      productId:{
8          type: String,
9          required: [true, ""],
10     },
11     title: {
12         type: String,
13         required: [true, ""],
14     },
15     description: {
16         type: String,
17         required: [true, ""],
18     },
19     price: {
20         type: String,
21         required: [true, ""],
22     },
23     image: {
24         type: String,
25         required: [true, ""],
26     }
27 }
28 const product = mongoose.model("product", productmodel);
29 module.exports = product;
```

User Model



```
EXPLORER
NODE-BACKEND
models
cart.js
orders.js
product.js
user.js
node_modules
uploads
.gitignore
app.js M
package-lock.json
package.json
TIMELINE
NPM SCRIPTS
```

user.js

```
models > user.js > usermodel > mobile
1  const mongoose = require('mongoose');
2  const usermodel = {
3      userId: {
4          type: String,
5          unique: true,
6          required: [true, ""],
7      },
8      fname: {
9          type: String,
10         // unique: true,
11         required: [true, "Please enter your name"],
12     },
13     lname: {
14         type: String,
15         // unique: true,
16         required: [true, "Please enter your name"],
17     },
18     email: {
19         type: String,
20         // unique: true,
21         required: [true, "Please enter your email"],
22     },
23     mobile: {
24         type: Number,
25         required: [true, "Please enter your mobile number"],
26         minlength: 10,
27         maxlength: 10,
28     },
29     password: {
30         type: String,
31         required: [true, "Please enter your password"],
32     },
33     role: {
34         type: String,
35         // required: [true, "Please enter your role"],
36         default: "user"
37     }
38 }
39 const user = mongoose.model("user", usermodel)
40 module.exports = user;
```

master* 0 0 Ln 23, Col 14 Spaces: 4 UTF-8 LF JavaScript Go Live

RESULTS & CONCLUSIONS

- Successfully completed implementation of a MongoDB + Express JS + Angular + Node JS (MEAN Stack) Application.
- Role based authentication is handled with Role Guards.
- Implemented of a modal window to edit products
- Gained in-depth knowledge about the working of an ecommerce store.

CODE

GITHUB LINK : https://github.com/aakar-mutha/Zing-Ecommerce