E-Commerce Application

Sri Vibhav J
CS20B047

Overview

The e-commerce application with the specified requirements is built using React.js, Node.js, MongoDB, MySQL. It is built with React on the front end, node js on the backend, mongoose connected with the local instance of mongodb and mysql module connected with the local instance of MySQL.

Features Implemented

Starting with the customer dashboard, I have implemented register customers and login for customers. I added the credentials (username and password) in MongoDB which is easy to use in web applications and session management. The rest of the data is stored in the MySQL database. The inventory database consists of many items being sold by different sellers. The home page of the dashboard features all the items in the inventory. A search bar is included which can search for items. A demo of the same is attached in the video.

An ‘add to cart’ button is included after every item displayed and the user can tune in the quantity parameter and add the item to the cart. Similarly, an option is provided to remove items from the cart. Along with the cart, a minimal banking interface is implemented, wherein all the banks that the user has accounts linked in will be displayed. The user can choose one bank and type out his/her account number and pin to continue with the payment. The user can also avail discounts by choosing a coupon out of a lot of them displayed. I made a decision to take out 1% of the transaction fee for the platform purpose. All the account’s balance will be updated after clicking on the checkout button. The customer’s balance reduces by the net amount and the advertiser's amount reduces by the discount increasing the seller’s account by total amount.

An ‘orders’ page is included where all the orders that are done by the user are displayed. This includes the order tracking status - dispatched from the seller, in warehouse, received, or returned. Before receiving the order, the customer is prompted for an OTP to receive his order. The OTP is stored in the database along with the order table elements. The order can be returned by the user. If done so, the order will be returned to the seller and the transaction will revert back.

The seller dashboard features the sold and returned items. The orders table is queried for the details of the seller’s items which are sold and returned. These items are rendered on the front end after an API call. This helps the seller to manage his inventory.

The warehouse dashboard displays two buttons to receive all and dispatch all the items in the inventory. I made this to simulate real world dispatch and receival of items in warehouses. The receive all button on click, will take in all the items that are sent to this warehouse. And the dispatch all button on click, sends all the items to the next place in the path from the warehouse. Accordingly the order status is updated in the seller and customer dashboard. Also, an inventory section is displayed in the dashboard showing all the items currently inside the warehouse.

Conclusion

The term project is made efficiently following the normalization norms and converting each table into its maximum normal form. The database design therefore is made modular and efficient. In the backend, all the API calls are written with SQL queries running after an API call. The data is split between MongoDB and MySQL, with usernames and passwords (session management is easy with mongo) stored in mongodb, and the remaining data (on which select, join operations must be used) getting into the MySQL database. MySQL is used for tables like accounts and orders whose data is to be consistent which is a main drawback of NoSQL systems. In any case that the data is needed to be scaled horizontally (columns to be added), the new attributes can be accommodated inside the MongoDB table of that respective entity, given the fact that NoSQL databases (especially document oriented) are flexible in such cases.