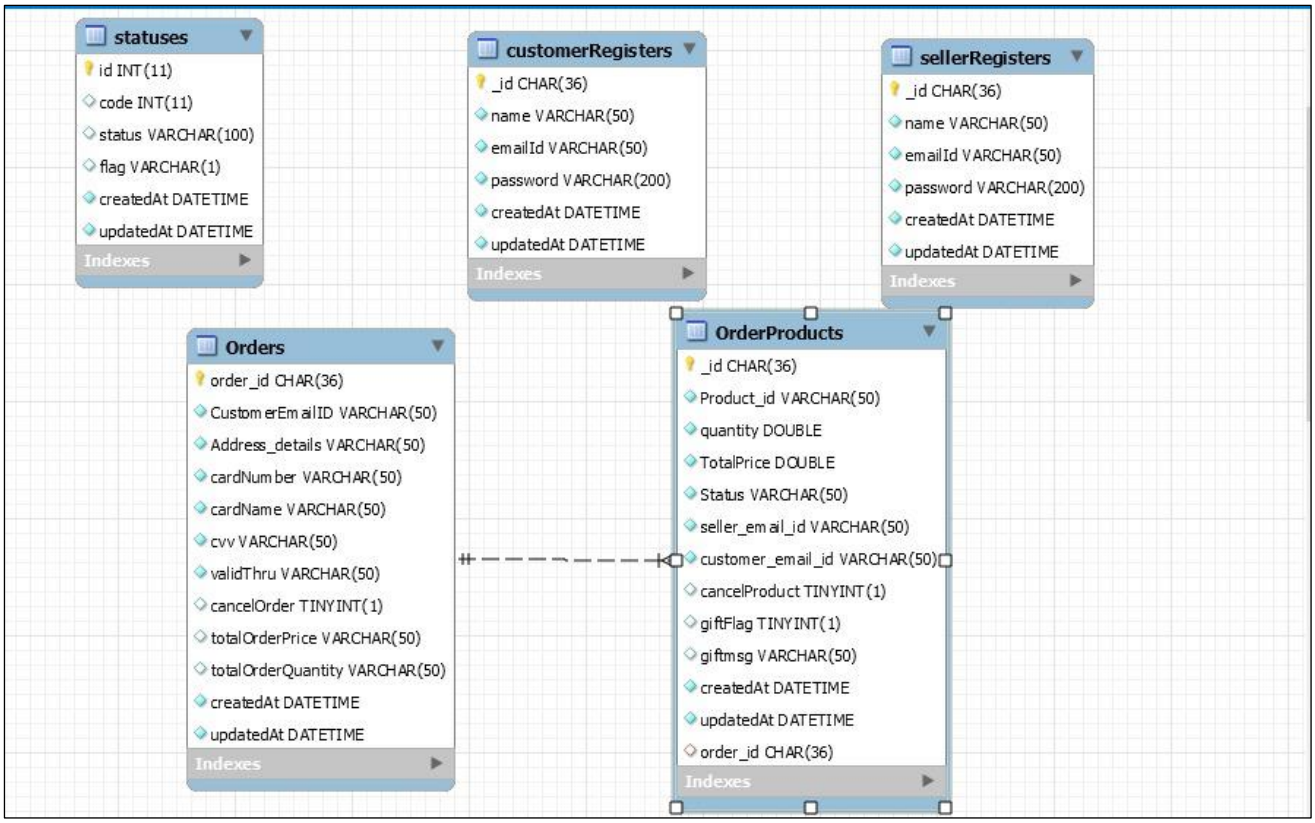
Database Schema

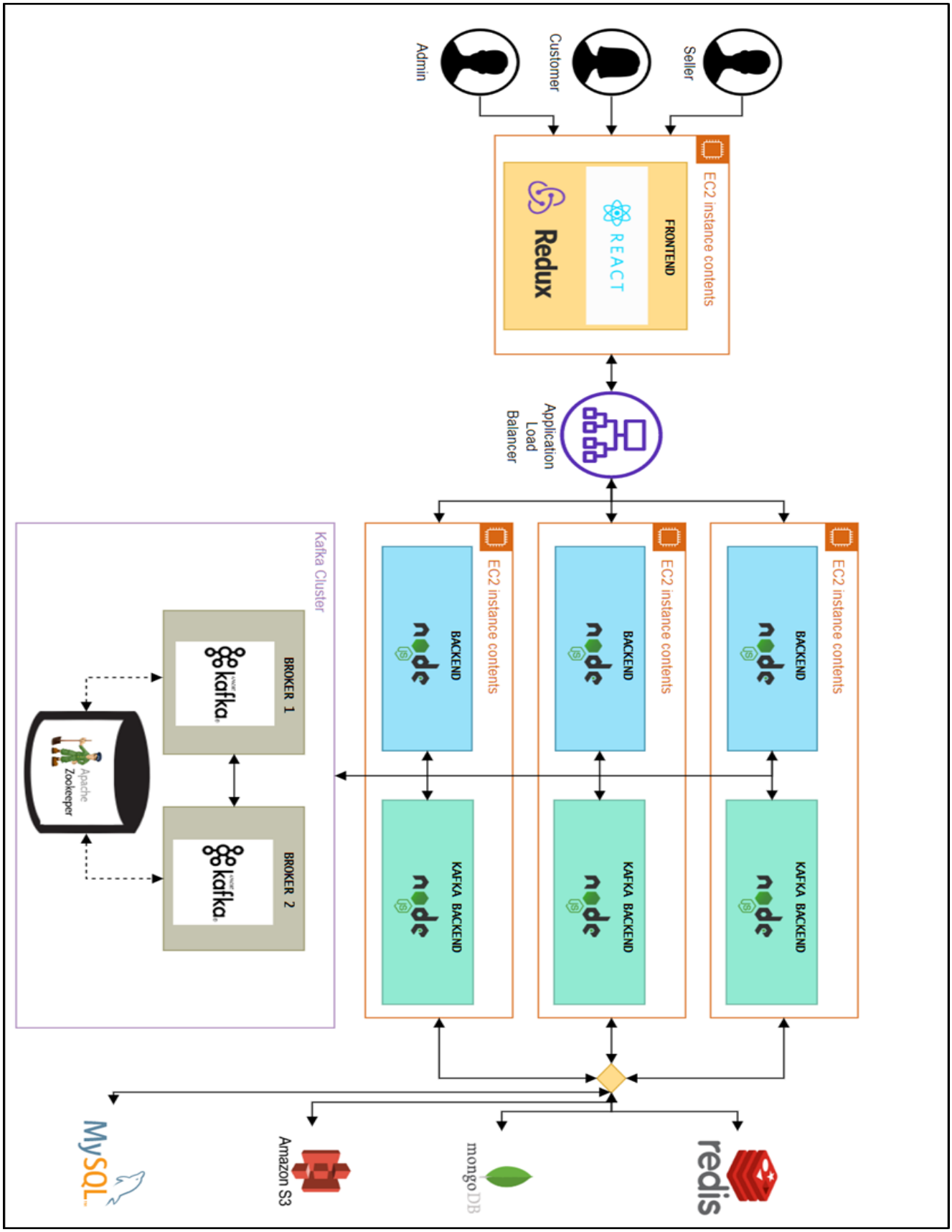


MySQL


sellerProfile
_id objectId NN
emailID string NN
name string NN
phone string
ProfilePictureUrl string
street string NN
city string NN
state string NN
country string NN
zipcode string NN
category
_id objectId NN
name string NN
productCount number NN
comment
_id objectId NN
customerId string NN
productId objectId NN
title string NN
comment string NN
rating number NN
product
_id objectId NN
sellerID objectId NN
validFlag string NN
sellerEmailId string NN
sellerName string NN
productName string NN
productCategory string NN
productPrice number NN
averageRating number 0.0
productDescription string NN
photos[] string
clickCount[{ }]
date string NN
count number NN
customer
_id objectId NN
emailId unique string NN
name string NN
phone string
profilePictureUrl string
city string
state string
savedProducts [ ]
productId string NN
sellerEmailId string NN
cartProducts [ ]
productId string NN
sellerEmailId string NN
quantity number NN
totalProductPrice number NN
giftFlag string false
giftMessage string
addresses [ ]
addressName string NN
street string NN
city string NN
state string NN
country string NN
zipcode string NN
phone string NN
paymentCards []
cardName string NN
cardNumber string NN
expirationDate date NN
cvv string NN

MongoDB

System Architecture and Design



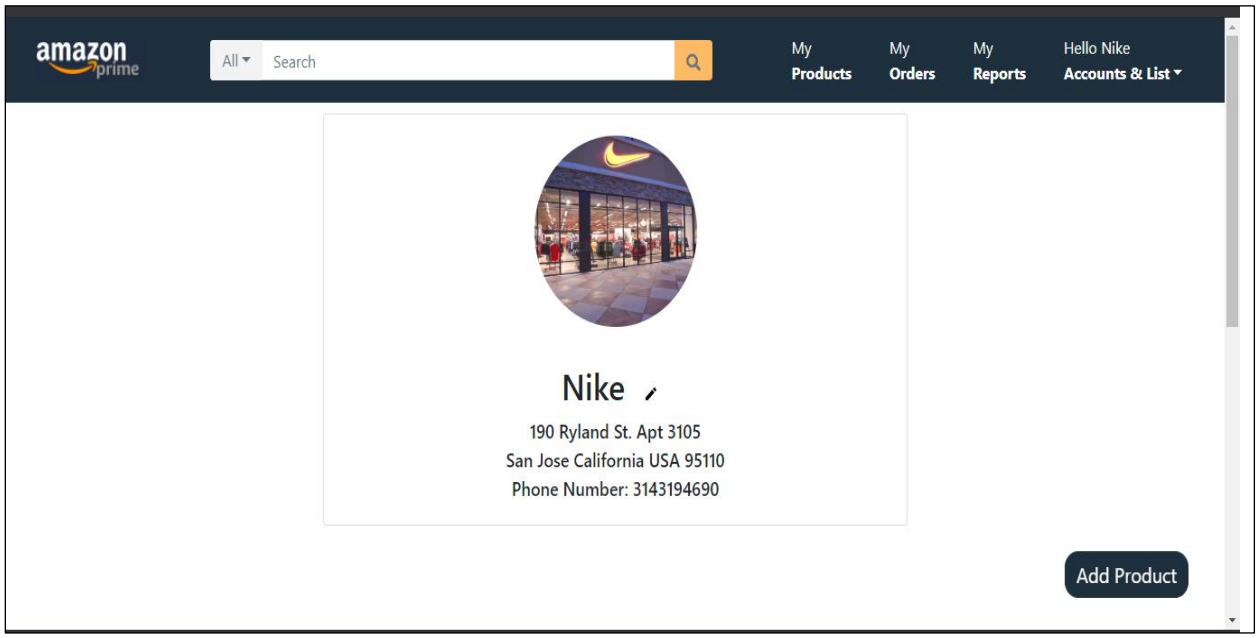
Object management policy

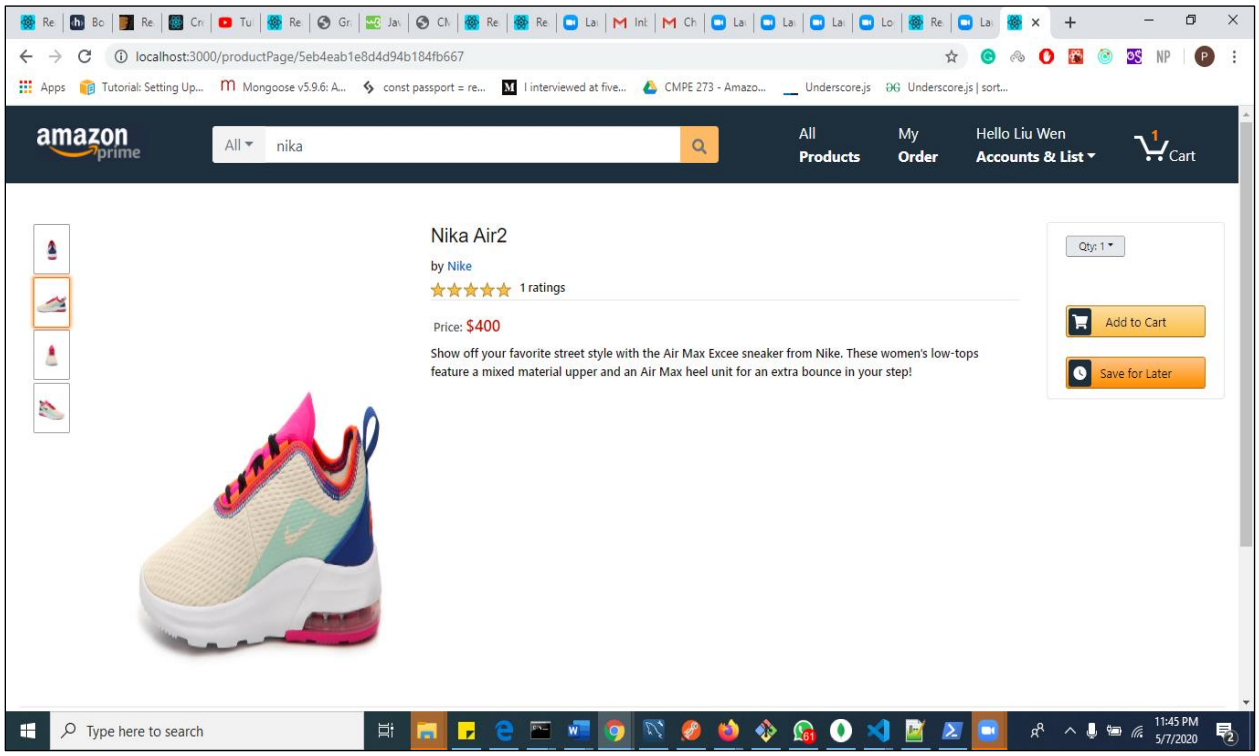
We are storing different types of objects in different file storage systems. The objects that are incoherent and do not need any relations to be fetched are stored in MongoDB. MongoDB also makes it quicker to fetch data by avoiding joins, making it read easy. All the important information is stored in MySQL to maintain ACID (Atomicity, Consistency, Isolation, and Durability) state. Large objects like pictures and images are stored in an Amazon S3 bucket and can be easily accessed by URL. All these are stored on different cloud instances and they are made to work synchronously without any duplication of data keeping the system in a single state.

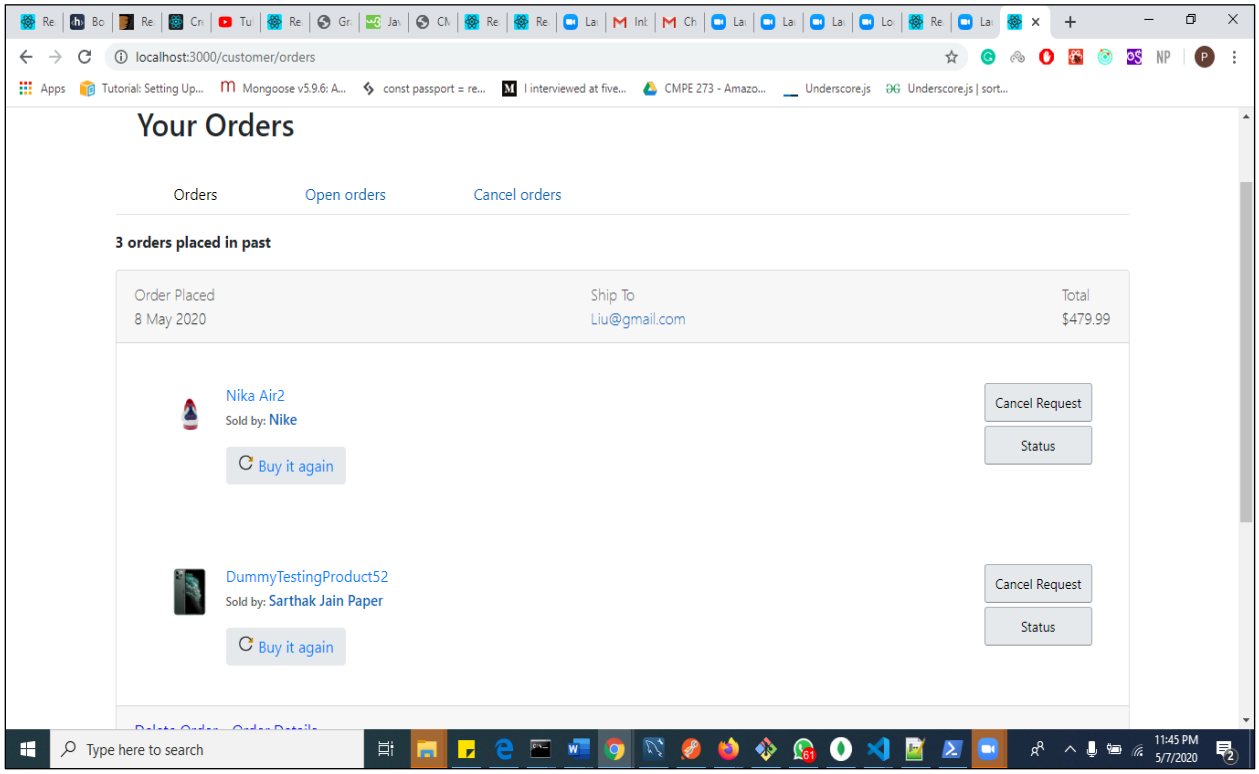
Handling heavy weight resources

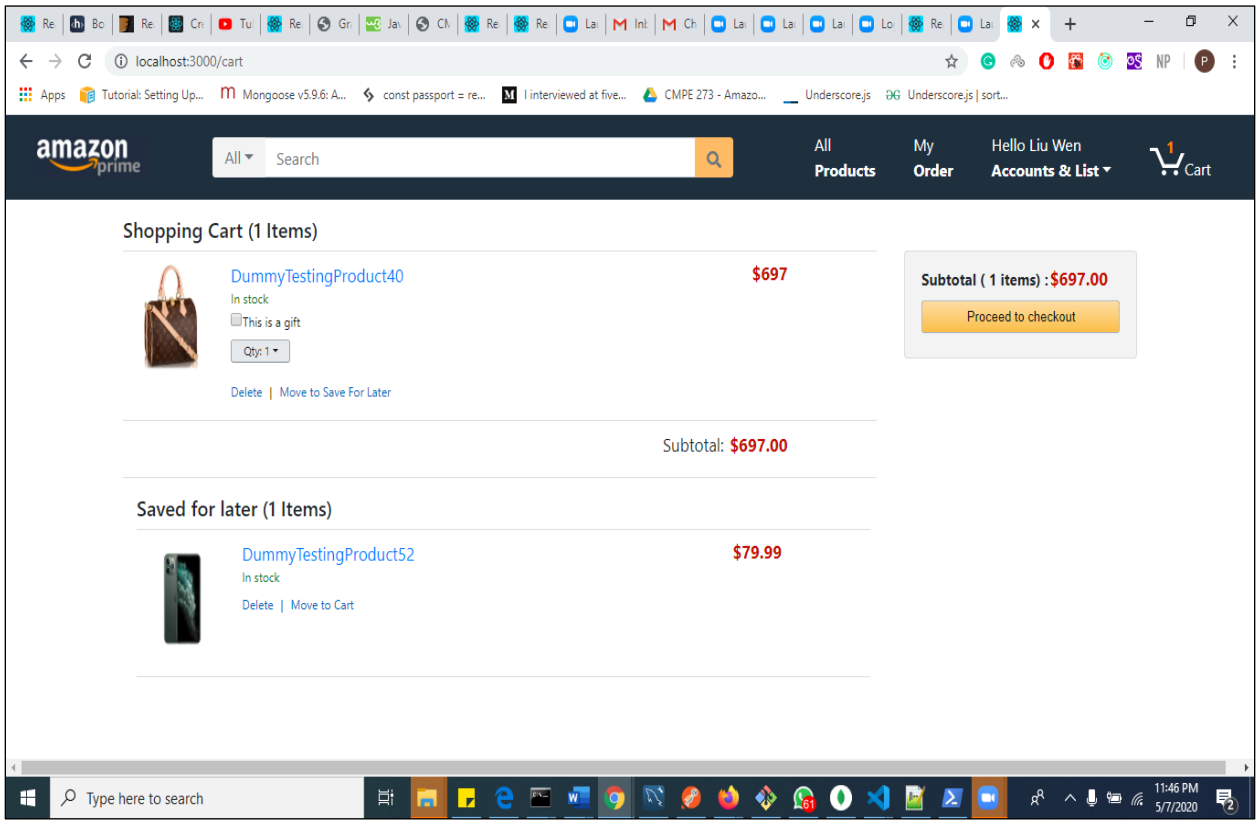
Firstly, we made the resources read easy and write intensive. Like in products collection we had denormalized a lot of columns, but it made it easier to fetch products and get related data. Similarly, the same policy was followed. Secondly, we used Redis cache to store large collections like products to make it faster to read. Finally, images were stored in AmazonS3 bucket instead of blob format in the DB, increasing the read times and improving efficiency.

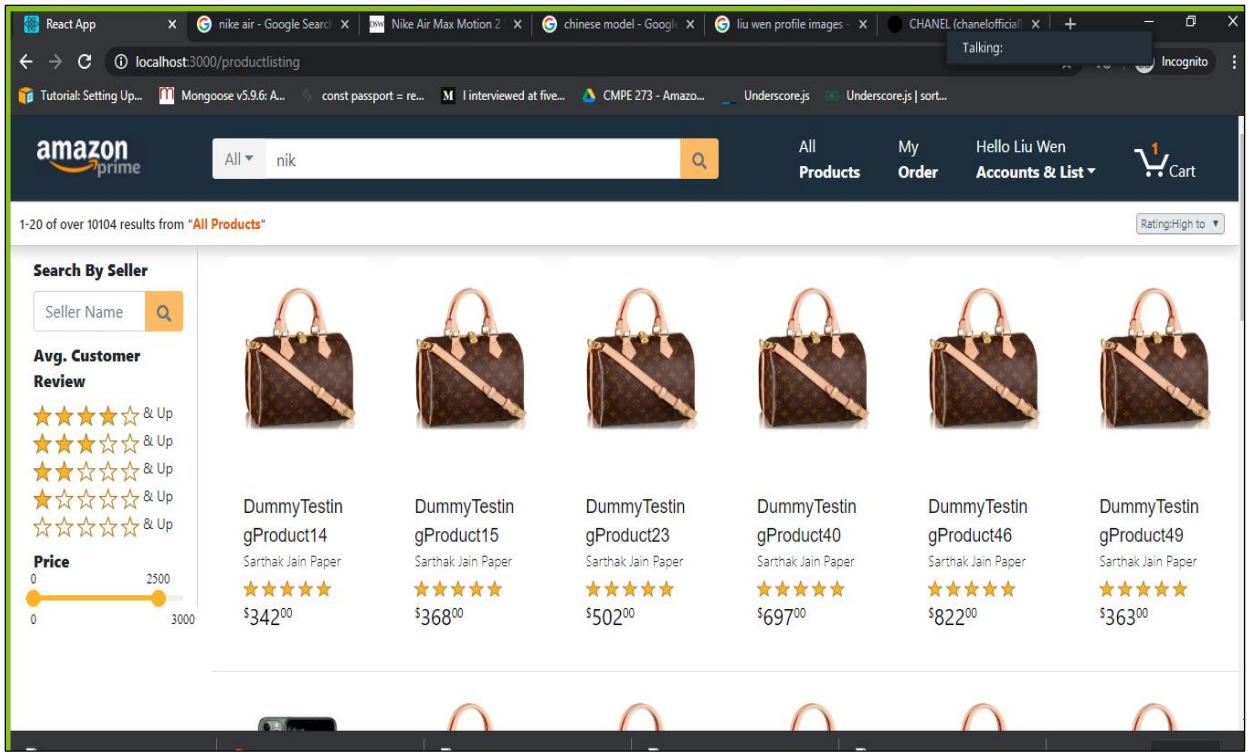
Application

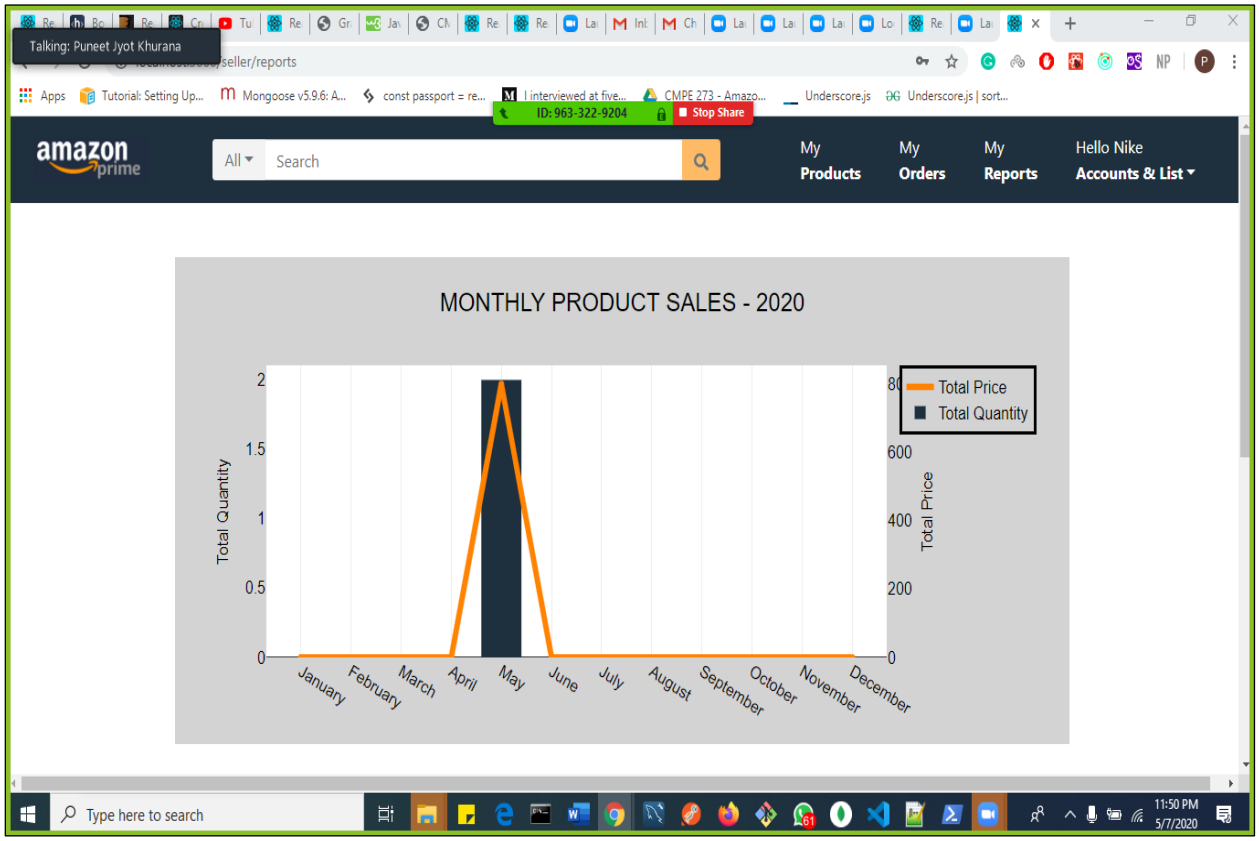


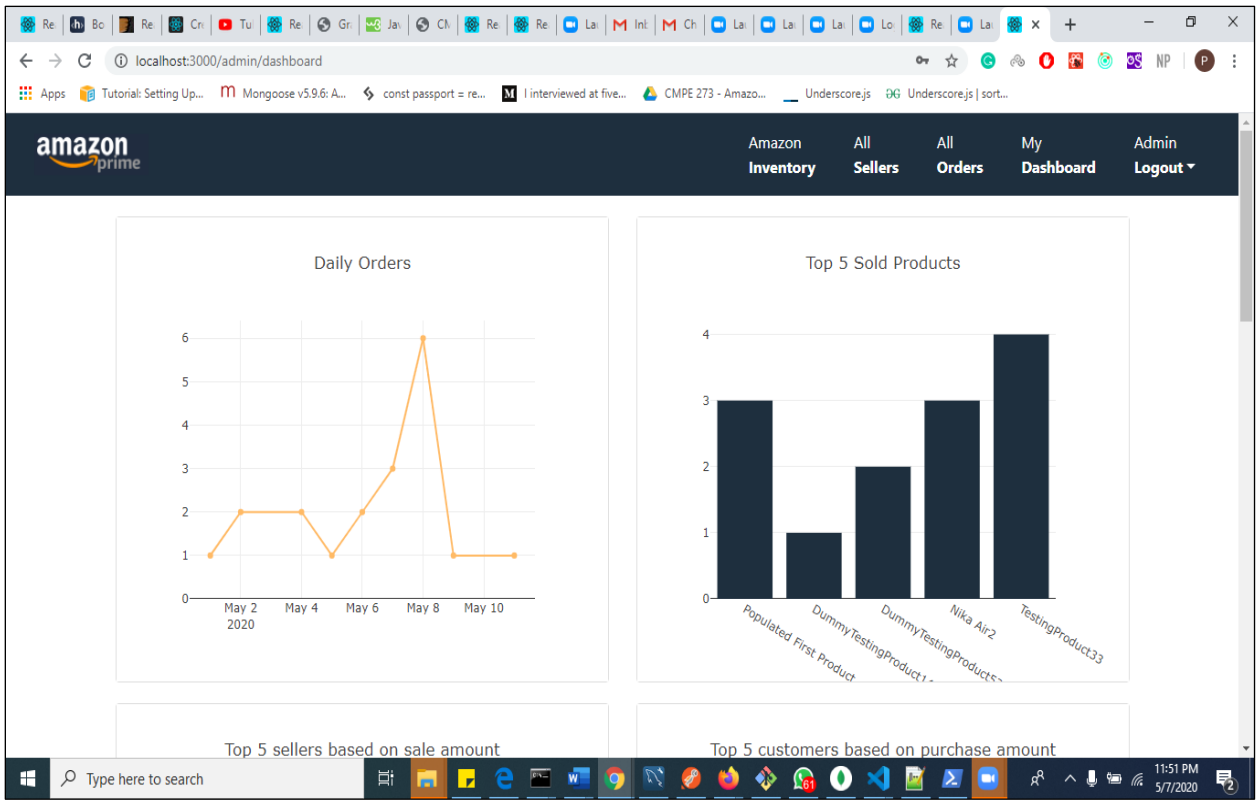


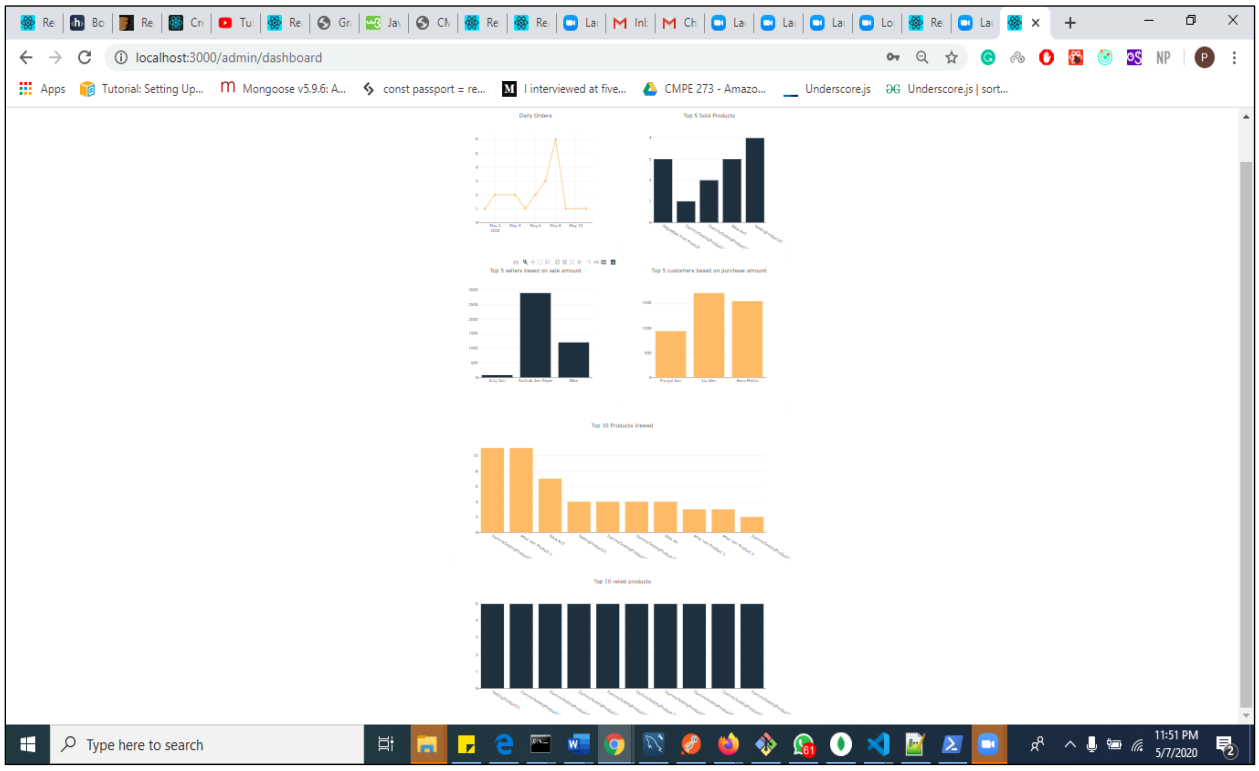

localhost:3000/customer/orders
Your Orders
Orders
Open orders
Cancel orders
3 orders placed in past
Order Placed
8 May 2020
Ship To
Liu@gmail.com
Total
$479.99
Nika Air2
Sold by: Nike
Buy it again
Cancel Request
Status
DummyTestingProduct52
Sold by: Sarthak Jain Paper
Buy it again
Cancel Request
Status

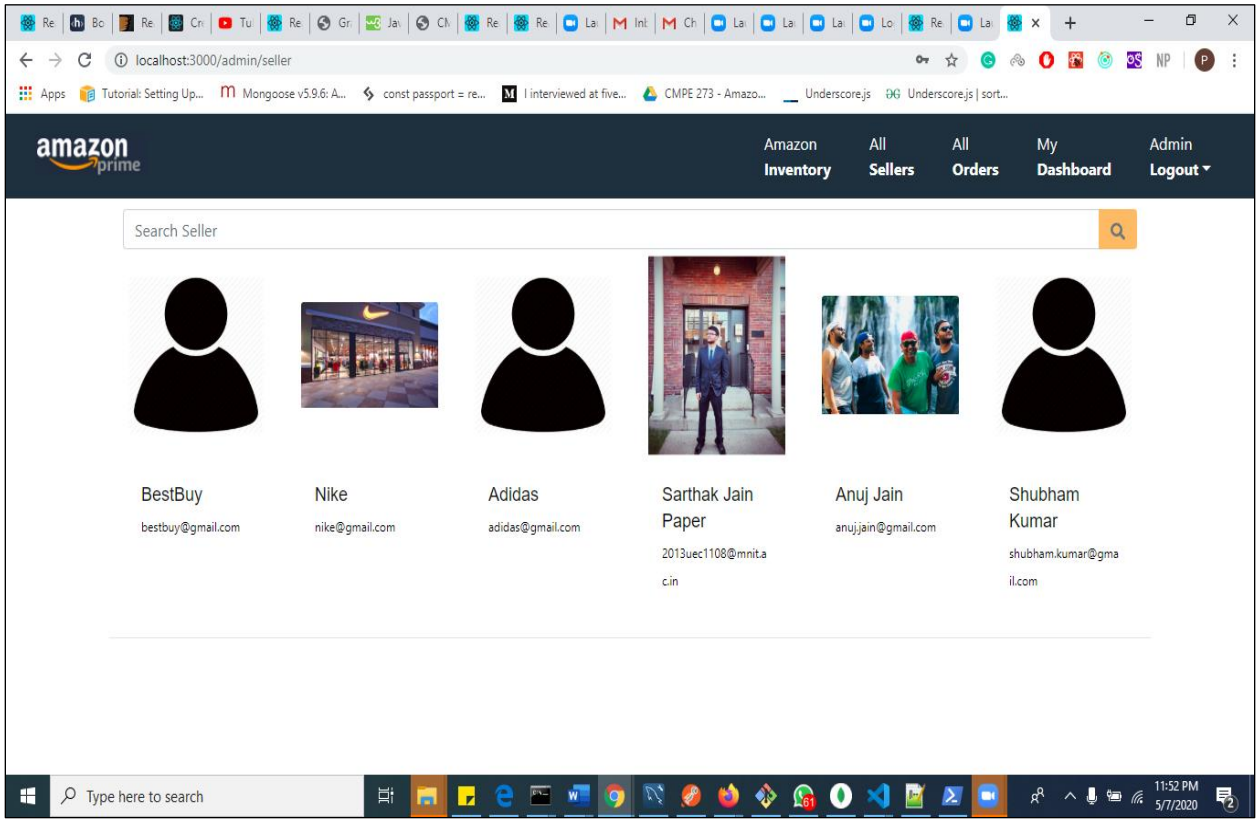

localhost:3000/cart
amazon prime
All
Search
All Products
My Order
Hello Liu Wen
Accounts & List
1
Cart
Shopping Cart (1 Items)
DummyTestingProduct40
In stock
This is a gift
Qty: 1
Delete | Move to Save For Later
$697
Subtotal (1 items) : $697.00
Proceed to checkout
Subtotal: $697.00
Saved for later (1 Items)
DummyTestingProduct52
In stock
Delete | Move to Cart
$79.99

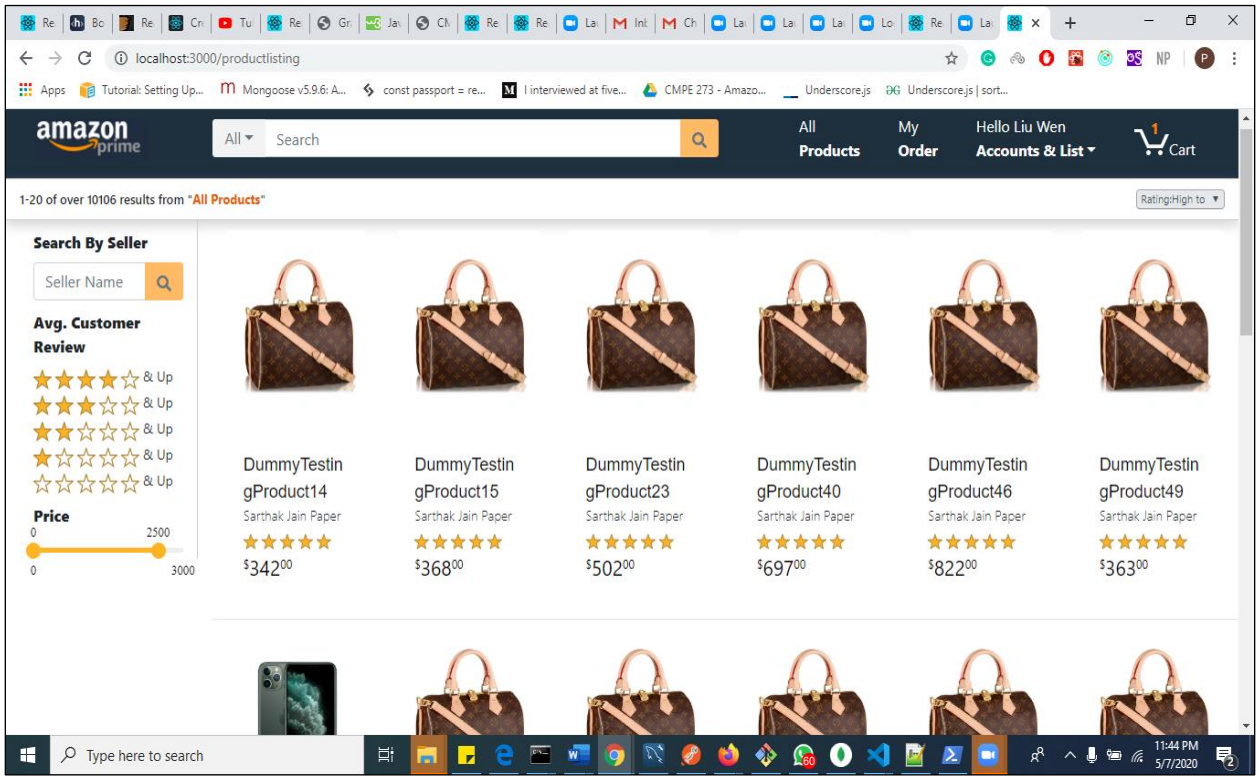

React App
localhost:3000/productlisting
amazon prime
All
nik
All Products
My Order
Hello Liu Wen
Accounts & List
Cart
1-20 of over 10104 results from "All Products"
Search By Seller
Seller Name
Avg. Customer Review
& Up
Price
0
2500
3000
DummyTestingProduct14
Sarthak Jain Paper
$342.00
DummyTestingProduct15
Sarthak Jain Paper
$368.00
DummyTestingProduct23
Sarthak Jain Paper
$502.00
DummyTestingProduct40
Sarthak Jain Paper
$697.00
DummyTestingProduct46
Sarthak Jain Paper
$822.00
DummyTestingProduct49
Sarthak Jain Paper
$363.00

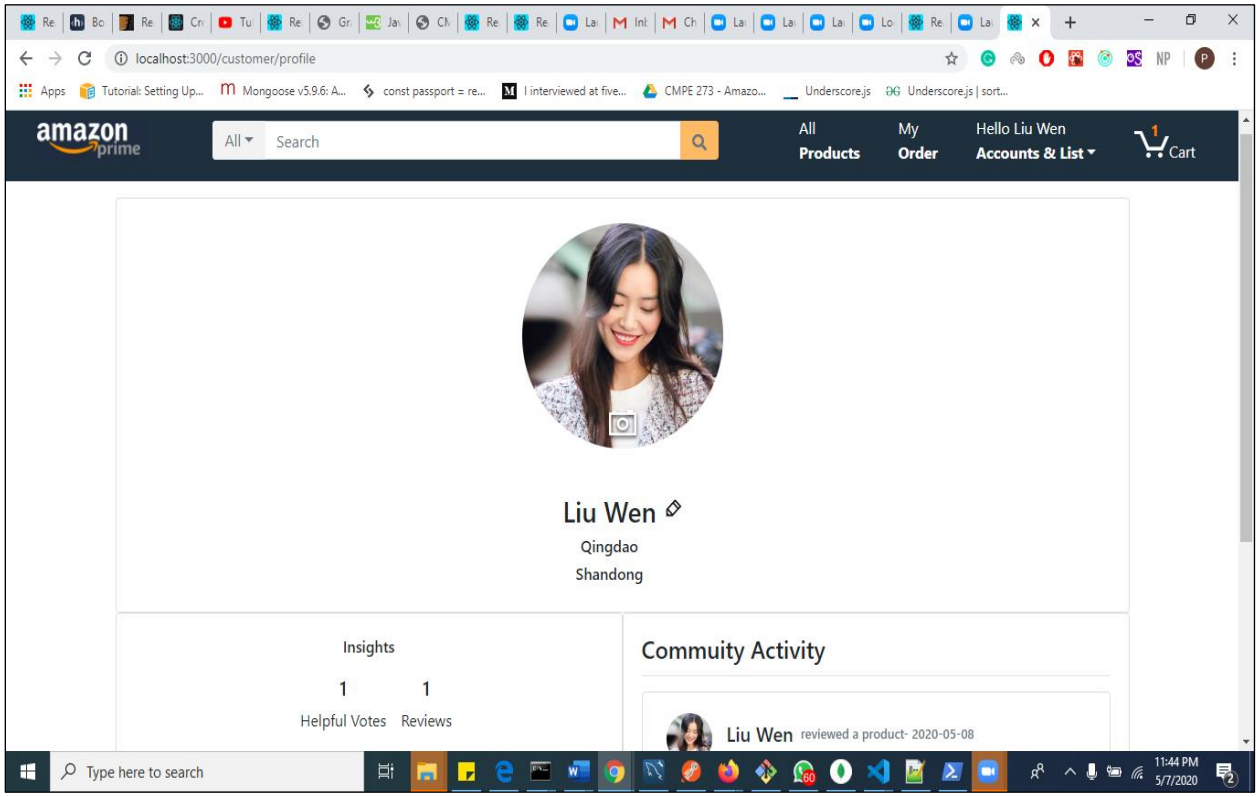

Talking: Puneet Jyot Khurana
seller/reports
ID: 963-322-9204
Stop Share
amazon prime
All
Search
My Products
My Orders
My Reports
Hello Nike
Accounts & List
MONTHLY PRODUCT SALES - 2020
Total Price
Total Quantity
Total Quantity
Total Price
0
0.5
1
1.5
2
200
400
600
800
January
February
March
April
May
June
July
August
September
October
November
December
Type here to search
11:50 PM
5/7/2020


localhost:3000/admin/dashboard
amazon prime
Amazon Inventory
All Sellers
All Orders
My Dashboard
Admin Logout
Daily Orders
May 2 2020
May 4
May 6
May 8
May 10
Top 5 Sold Products
Populated First Product
DummyTestingProduct1
DummyTestingProduct2
Nika Air2
TestingProduct33
Top 5 sellers based on sale amount
Top 5 customers based on purchase amount
Type here to search
11:51 PM
5/7/2020


localhost:3000/admin/dashboard
Type here to search
11:51 PM
5/7/2020


localhost:3000/admin/seller
amazon prime
Amazon Inventory
All Sellers
All Orders
My Dashboard
Admin Logout
Search Seller
BestBuy
bestbuy@gmail.com
Nike
nike@gmail.com
Adidas
adidas@gmail.com
Sarthak Jain Paper
2013uec1108@mnit.ac.in
Anuj Jain
anujjain@gmail.com
Shubham Kumar
shubham.kumar@gmail.com


localhost:3000/productlisting
amazon prime
All
Search
All Products
My Order
Hello Liu Wen
Accounts & List
Cart
1-20 of over 10106 results from "All Products"
Rating:High to
Search By Seller
Seller Name
Avg. Customer Review
& Up
Price
0
2500
3000
DummyTestingProduct14
Sarthak Jain Paper
$342.00
DummyTestingProduct15
Sarthak Jain Paper
$368.00
DummyTestingProduct23
Sarthak Jain Paper
$502.00
DummyTestingProduct40
Sarthak Jain Paper
$697.00
DummyTestingProduct46
Sarthak Jain Paper
$822.00
DummyTestingProduct49
Sarthak Jain Paper
$363.00


localhost:3000/customer/profile
amazon prime
All
Search
All Products
My Order
Hello Liu Wen
Accounts & List
1 Cart
Liu Wen
Qingdao
Shandong
Insights
1
Helpful Votes
1
Reviews
Commuity Activity
Liu Wen reviewed a product- 2020-05-08
Type here to search
11:44 PM
5/7/2020

Server implementation for Entity Objects

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const category = new Schema({
  name:{type:String,required:true},
  productCount:{type:Number,default:0},
},{
  timestamps:true
});

module.exports = mongoose.model('category',category);
```

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const clickCount = new Schema({
  productId:{type: mongoose.Schema.Types.ObjectId, ref: "product"},
  productName:{type:String,required:true},
  lastViewedCustomDate:{type:String,default:""},
  count:{type:Number,default:0}
},{
  timestamps:true
});

module.exports = mongoose.model('clickCount',clickCount);
```

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
//const ObjectId = Schema.Types.ObjectId;

const comment = new Schema(
  {
    customerId: { type: String, required: true },
    productId: { type: mongoose.Schema.ObjectId, required: true },
    title: { type: String, required: true },
    comment: { type: String, required: true },
    rating: { type: Number, required: true },
  },
  {
    timestamps: true,
  }
);

module.exports = mongoose.model("comment", comment);
```

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const uniqueValidator = require("mongoose-unique-validator");

const savedProduct = new Schema(
  {
    productId: { type: String, required: true },
    sellerEmailId: { type: String, required: true }
  },
  {
    timestamps: true
  }
);

const cartProduct = new Schema(
  {
    productId: { type: String, required: true },
    sellerEmailId: { type: String, required: true },
    quantity: { type: Number, required: true, default: 1 },
    totalProductPrice: {type: Number, required: true},
    giftFlag: { type: String, default: "false" },
    giftMessage: { type: String, default: "" }
  },
  {
    timestamps: true
  }
);

const address = new Schema(
  {
    addressName: { type: String, required: true },
    street: { type: String, required: true },
    city: { type: String, required: true },
    state: { type: String, required: true },
    country: { type: String, required: true },
    zipcode: { type: String, required: true },
    phone: { type: String, required: true }
  },
  {
    timestamps: true
  }
);

const card = new Schema(
  {
    cardName: { type: String, required: true },
    cardNumber: { type: String, required: true },
    expirationDate: { type: Date, required: true },
```

```
    cvv: { type: String, required: true }
  },
  {
    timestamps: true
  }
);

const customerProfile = new Schema(
  {
    emailId: { type: String, required: true, unique: true },
    name: { type: String, required: true },
    phone: { type: String, default: "" },
    profilePictureUrl: { type: String, default: "default.png" },
    city: { type: String, default: "" },
    state: { type: String, default: "" },
    savedProducts: [savedProduct],
    cartProducts: [cartProduct],
    addresses: [address],
    paymentCards: [card]
  },
  {
    timestamps: true
  }
);
customerProfile.plugin(uniqueValidator);

module.exports = mongoose.model("customer", customerProfile);
```

```
const Sequelize = require("sequelize");
const connection = require("../connections/sequelize_connection");

const Dt = Sequelize.DataTypes;

//cutsomer registration model
const order = {
  order_id: {
    type: Dt.UUID,
    primaryKey: true,
    allowNull: false,
    defaultValue: Dt.UUIDV1
  },
  CustomerEmailID: {
    type: Dt.STRING(50),
    allowNull: false
  },

  Address_details: {
    type: Dt.STRING(50),
    allowNull: false
```

```
  },
  cardNumber: {
    type: Dt.STRING(50),
    allowNull: false
  },
  cardName: {
    type: Dt.STRING(50),
    allowNull: false
  },
  cvv: {
    type: Dt.STRING(50),
    allowNull: false
  },
  validThru: {
    type: Dt.STRING(50),
    allowNull: false
  },
  cancelOrder: {
    type: Dt.BOOLEAN,
    defaultValue: false
  },
  totalOrderPrice: {
    type: Dt.STRING(50)
  },
  totalOrderQuantity: {
    type: Dt.STRING(50)
  }
};
const Order = connection.define("Order", order);

//cutsomer registration model
const orderproduct = {
  _id: {
    type: Dt.UUID,
    primaryKey: true,
    allowNull: false,
    defaultValue: Dt.UUIDV1
  },
  Product_id: {
    type: Dt.STRING(50),
    allowNull: false
  },
  quantity: {
    type: Dt.DOUBLE,
    allowNull: false
  },
  TotalPrice: {
    type: Dt.DOUBLE,
    allowNull: false
  },
  Status: {
    type: Dt.STRING(50),
```

```
    allowNull: false
  },
  seller_email_id: {
    type: Dt.STRING(50),
    allowNull: false
  },
  customer_email_id: {
    type: Dt.STRING(50),
    allowNull: false
  },
  cancelProduct: {
    type: Dt.BOOLEAN,
    defaultValue: false
  },
  giftFlag: {
    type: Dt.BOOLEAN,
    defaultValue: false
  },
  giftmsg: {
    type: Dt.STRING(50)
  }
};
const OrderProduct = connection.define("OrderProduct", orderproduct);

//static status table
const statusSchema = {
  code: { type: Dt.INTEGER },
  status: { type: Dt.STRING(100) },
  flag: { type: Dt.STRING(1) }
};
const status = connection.define("status", statusSchema);

connection.sync();

OrderProduct.belongsTo(Order, { foreignKey: "order_id" });

module.exports = {
  Order,
  OrderProduct,
  status
};
```

```
const mongoose = require('mongoose');
const mongoosePaginate = require('mongoose-paginate');
const Schema = mongoose.Schema;

// date in clickCount can be mm/dd/yyy. Before insertion or updation, get your da
te in this format?
const clickCount = new Schema({
```

```
  date:{type:String,required:true},
  count:{type:Number,required:true}
},{
  timestamps:true
});

const product = new Schema({
  sellerId:{type: mongoose.Schema.Types.ObjectId, ref: "seller"},
  validFlag:{type:String,default:"true"},
  sellerEmailId:{type:String,required:true},
  sellerName:{type:String,required:true},
  productName:{type:String,required:true},
  productCategory:{type:String,required:true},
  productPrice:{type:Number,required:true},
  averageRating:{type:Number,default:0.0},
  productDescription:{type:String,required:true},
  photos:[String],
  clickCount:[clickCount],
},{
  timestamps:true
});

product.plugin(mongoosePaginate);

module.exports = mongoose.model('product',product);
```

```
const Sequelize = require("sequelize");
const bcrypt = require("bcrypt");
const connection = require("../connections/sequelize_connection");

const Dt = Sequelize.DataTypes;

//cutsomer registration model
const cust_register = {
  _id: {
    type: Dt.UUID,
    primaryKey: true,
    allowNull: false,
    defaultValue: Dt.UUIDV1
  },
  name: {
    type: Dt.STRING(50),
    allowNull: false
  },
  emailId: {
    type: Dt.STRING(50),
    allowNull: false,
    unique: true
  },
  password: { type: Dt.STRING(200), allowNull: false }
```

```
};
const customerRegister = connection.define("customerRegister", cust_register);

//cutsomer registration model
const seller_register = {
  _id: {
    type: Dt.UUID,
    primaryKey: true,
    allowNull: false,
    defaultValue: Dt.UUIDV1
  },
  name: {
    type: Dt.STRING(50),
    allowNull: false
  },
  emailId: {
    type: Dt.STRING(50),
    allowNull: false,
    unique: true
  },
  password: { type: Dt.STRING(200), allowNull: false }
};
const sellerRegister = connection.define("sellerRegister", seller_register);

connection.sync();

module.exports = {
  customerRegister,
  sellerRegister
};
```

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const uniqueValidator = require("mongoose-unique-validator");

const sellerProfile = new Schema(
  {
    emailId: { type: String, required: true, unique: true },
    name: { type: String, required: true },
    phone: { type: String, default:"" },
    profilePictureUrl: { type: String, default: "default.png" },
    street: { type: String, default:"" },
    city: { type: String, default:"" },
    state: { type: String, default:"" },
    country: { type: String, default:"" },
    zipcode: { type: String, default:"" }
  },
  {
    timestamps: true
```

```
  }
);
sellerProfile.plugin(uniqueValidator);

module.exports = mongoose.model("seller", sellerProfile);
```

Server implementation of Security and Session Objects

```
"use strict";
var JwtStrategy = require("passport-jwt").Strategy;
var ExtractJwt = require("passport-jwt").ExtractJwt;
const passport = require("passport");
const Config = require("./config");
const kafka = require("./kafka/client");

// Setup work and export for the JWT passport strategy
function auth() {
  var opts = {
    jwtFromRequest: ExtractJwt.fromAuthHeaderWithScheme("jwt"),
    secretOrKey: Config.secret
  };
  passport.use(
    new JwtStrategy(opts, (jwt_payload, callback) => {
      console.log('JWT_Payload: ');
      console.log(jwt_payload);
      var msg = {};
      msg.user_id = jwt_payload._id;
      msg.userRole = jwt_payload.category;
      console.log("msg in backend: ");
      console.log(msg);
      kafka.make_request("passport", msg, function(err, results) {
        if (err) {
          return callback(err, false);
        }
        if (results) {
          callback(null, results);
        } else {
          callback(null, false);
        }
      });
    })
  );
}

exports.auth = auth;
exports.checkAuth = passport.authenticate("jwt", { session: false });
```

Main Server Code

Frontend

```
import React from "react";
import topNav from "./navbar";
import CustomerProfile from "./customer/profile/CustomerProfile";
import PaymentAndAddressPage from "./customer/profile/PaymentAndAddressPage";

import SellerProfile from "./seller/profile/SellerProfile";
import SellerOrderPage from "./seller/order/OrderPage";
import SellerCancelledOrder from "./seller/order/CancelledOrder";
import OpenSellerOrder from "./seller/order/OpenSellerOrder";
import AdminOrder from "./admin/order/OrderPage";
import { Route } from "react-router-dom";
import { connect } from "react-redux";
//import cookie from 'react-cookies';
import Login from "./Login/Login";
import RegisterCustomer from "./Register/RegisterCustomer";
import RegisterSeller from "./Register/RegisterSeller";
import ProductList from "./products/productsList";
import ProductPage from "./products/productPage/ProductPage";
import OrderPage from "./customer/order/OrderPage";
import CancelledOrder from "./customer/order/CancelledOrder";
import OpenOrder from "./customer/order/OpenOrder";
import CartAndSaved from "./customer/cartAndSaved/CartAndSaved";
import CategoryList from "./admin/category/CategoryList";
import SellerList from "./admin/seller/SellerList";
import SelectAddressAndPayment from "./customer/checkout/SelectAddressAndPayment"
;
import SellerReport from "./seller/reports/sellerReportContainer";
import Dashboard from "./admin/dashboard/Dashboard";

class bodyCont extends React.Component {
  render() {
    return (
      <div>
        <Route path='/' component={topNav} />
        <Route path='/customer/profile' component={CustomerProfile} />
        <Route path='/seller/profile' component={SellerProfile} />
        <Route path='/seller/order' component={SellerOrderPage} />

        <Route path='/login' component={Login} />
        <Route path='/registerCustomer' component={RegisterCustomer} />
        <Route path='/registerSeller' component={RegisterSeller} />
        <Route path='/addressandpayment' component={PaymentAndAddressPage} />
        <Route path='/customer/orders' component={OrderPage} />
        <Route path='/admin/inventory' component={CategoryList} />
        <Route path='/admin/seller' component={SellerList} />
```

```
        <Route path='/productlist' component={ProductList} />
        <Route path='/productlisting' component={ProductList} />
        {/* <Route path="/paymentcard" component={PaymentCard} /> */}
        <Route path='/productPage/:id' component={ProductPage} />
        <Route path='/cart' component={CartAndSaved} />
        <Route path='/checkout' component={SelectAddressAndPayment} />
        <Route
          path='/customer/order/cancelledorders'
          component={CancelledOrder}
        />
        <Route
          path='/seller/cancelledorders'
          component={SellerCancelledOrder}
        />
        <Route path='/customer/order/openorders' component={OpenOrder} />
        <Route path='/seller/openorders' component={OpenSellerOrder} />
        <Route path='/admin/order' component={AdminOrder} />
        <Route path='/admin/dashboard' component={Dashboard} />
        <Route path='/seller/reports' component={SellerReport} />
      </div>
    );
  }
}

const mapStateToProps = (state) => {
  return {
    getType: state.getType,
  };
};

export default connect(mapStateToProps)(bodyCont);
```

Backend

```
// inbuilt package imports
const express = require("express");
const bodyParser = require("body-parser");
const session = require("express-session");
const cookieParser = require("cookie-parser");
const cors = require("cors");

// User defined module imports
const Config = require("./config");
const sellerProfile = require("./routes/Seller/profile");
const sellerProduct = require("./routes/Seller/product");
const sellerOrder = require("./routes/Seller/order");
const customerProduct = require("./routes/customer/product");
const adminProduct = require("./routes/admin/product");
const adminSeller = require("./routes/admin/seller");
```

```
const healthCheck = require("./routes/HealthCheck/healthCheck");

const app = express();
// setting view engine
app.set("view engine", "ejs");
// use body parser to parse JSON and urlencoded request bodies
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
// use cookie parser to parse request headers
app.use(cookieParser());
// use session to store user data between HTTP requests
app.use(
  session({
    secret: "sarthak_amazon_secure_string",
    resave: false,
    saveUninitialized: false,
    duration: 60 * 60 * 1000, // Overall duration of Session : 30 minutes : 1800
seconds
    activeDuration: 5 * 60 * 1000
  })
);
app.use(
  cors({
    origin: `${Config.applicationAddress}:${Config.applicationPort}`,
    credentials: true
  })
);

app.use(express.static("./ProfilePictures/Seller"));
app.use(express.static("./ProfilePictures/Customer"));
app.use(express.static("./ProfilePictures/Common"));

app.use("/healthCheck", healthCheck);
app.use("/login", require("./routes/account/login"));
app.use("/registerCustomer", require("./routes/account/registerCustomer"));
app.use("/registerSeller", require("./routes/account/registerSeller"));
app.use("/customer/profile", require("./routes/customer/profile"));
app.use("/seller/profile", sellerProfile);
app.use("/seller/orders", sellerOrder);
app.use("/product/customer", customerProduct);
app.use("/product/seller", sellerProduct);
app.use("/product/admin", adminProduct);
app.use("/admin/seller", adminSeller);
app.use("/customer/payment", require("./routes/customer/payment"));
app.use("/customer/address", require("./routes/customer/address"));
app.use("/customer/orders", require("./routes/customer/order/order"));
app.use("/admin/orders", require("./routes/admin/order"));

app.use(
  "/customer/cartProducts",
  require("./routes/customer/savedAndCartProducts")
);
```

```
app.use("/customer/checkout", require("./routes/customer/checkout"));
app.use("/product/status", require("./routes/tracking/tracking"));
app.use("/seller/analytics", require("./routes/reports/sellerReport"));
app.use("/admin/analytics", require("./routes/reports/adminReport"));

const server = app.listen(3001, () => {
  console.log("Server listening on port 3001");
});
```

Kafka Backend

```
//const Database=require('./Database');
var connection = new require("./kafka/Connection");
var connection_string = new require("./config");

//passport service
const passportService = require("./services/passport");
//account services
const accountService = require("./services/account");
//seller profile services
const sellerProfileService = require("./services/Seller/Profile");
//customer profile service
const customerProfileService = require("./services/customer/profile");
//customer product service
const customerProductService = require("./services/customer/Product");
//seller product service
const sellerProductService = require("./services/Seller/Product");
//seller order service
const sellerOrderService = require("./services/Seller/Order");

//admin product service
const adminProductService = require("./services/admin/Product");
//admin seller service
const adminSellerService = require("./services/admin/Seller");
//customer Payment service
const customerPaymentService = require("./services/customer/payment");
//customer Address service
const customerAddressService = require("./services/customer/address");
const orderAddressService = require("./services/customer/orders");
//saved and Cart Product Service
const savedAndCartProductService = require("./services/customer/savedAndCartProdu
cts");
//checkout Service
const checkoutService = require("./services/customer/checkout");
//tracking Service
const trackingService = require("./services/tracking");
//admin order service
const adminOrderServices =require("./services/admin/Orders")
//report Service
```

```
const reportService = require("./services/reports");
//connect to MongoDB
const Mongoose = require("mongoose");
var options = {
  useNewUrlParser: true,
  useUnifiedTopology: true
  //   ,
  //   reconnectInterval: 500,
  //   poolSize: 50,
  //   bufferMaxEntries: 0
};
Mongoose.connect(connection_string.connection_string, options)
  .then(() => console.log("Connected to MongoDB"))
  .catch(err => {
    console.log("Failed to connect to MongoDB");
    console.log(err);
  });

//handle topic's request
function handleTopicRequest(topic_name, fname) {
  //var topic_name = 'root_topic';
  var consumer = connection.getConsumer(topic_name);
  var producer = connection.getProducer();
  console.log("Kafka server is running ");
  consumer.on("message", function(message) {
    console.log("message received for " + topic_name);
    console.log(JSON.stringify(message.value));
    var data = JSON.parse(message.value);

    // Handling the make request that was called from backend server here in this
 function.
    fname.handle_request(data.data, function(err, res) {
      console.log("after handle: " + JSON.stringify(err));
      var result;
      if (err) {
        result = err;
      } else {
        result = res;
      }
      var payloads = [
        {
          topic: data.replyTo,
          messages: JSON.stringify({
            correlationId: data.correlationId,
            data: result
          }),
          partition: 0
        }
      ];
      producer.send(payloads, function(err, data) {
        console.log(data);
      });
```

```
      return;
    });
  });
}

//topics
handleTopicRequest("accounts", accountService);
handleTopicRequest("passport", passportService);
handleTopicRequest("sellerProfileService", sellerProfileService);
handleTopicRequest("customerProfile", customerProfileService);
handleTopicRequest("customerProductService", customerProductService);
handleTopicRequest("sellerProductService", sellerProductService);
handleTopicRequest("sellerOrderService", sellerOrderService);
handleTopicRequest("adminProductService", adminProductService);
handleTopicRequest("adminSellerService", adminSellerService);
handleTopicRequest("customerPaymentService", customerPaymentService);
handleTopicRequest("customerAddressService", customerAddressService);
handleTopicRequest("orderAddressService", orderAddressService);
handleTopicRequest("savedAndCartProductService", savedAndCartProductService);
handleTopicRequest("checkoutService", checkoutService);
handleTopicRequest("trackingService", trackingService);
handleTopicRequest("adminOrderServices", adminOrderServices);

handleTopicRequest("reportService", reportService);
```

Database Access

MySQL

```
const mysql = require("mysql2");
const credentials = require("./../config");

// create the connection to database
var mysql_connection = mysql.createPool({
  connectionLimit: 10,
  host: credentials.host,
  port: "3306",
  user: "admin",
  password: credentials.password,
  database: "amazon",
  dateStrings: true
});

mysql_connection.query("SET FOREIGN_KEY_CHECKS=0", (err, res) => {
  if (err) console.log("DB connection failed!!!");
  else {
    console.log("DB connection successful!!!");
  }
});
```

```
module.exports = mysql_connection;
```

Sequelize

```
const { Sequelize, DataTypes } = require("sequelize");
const credentials = require("./../config");
console.log(credentials);
const seq_connection = new Sequelize("amazon", "admin", credentials.password, {
  host: credentials.host,
  dialect: "mysql"
});

var test = seq_connection
  .authenticate()
  .then(function() {
    console.log("CONNECTED! ");
  })
  .catch(function(err) {
    console.log(err);
  })
  .done();
// const mysql = require("mysql");

// create the connection to database
// var connection = mysql.createPool({
//   connectionLimit: 10,
//   host: "mehtak.cimijgbx7bue.us-east-2.rds.amazonaws.com",
//   port: "3306",
//   user: "admin",
//   password: "admin#123",
//   database: "MEHTAK",
//   dateStrings: true
// });

// connection.query("SET FOREIGN_KEY_CHECKS=0", (err, res) => {
//   if (err) console.log("DB connection failed!!!");
//   else {
//     console.log("DB connection successful!!!");
//   }
// });
module.exports = seq_connection;
```

MongoDB

```
const applicationAddress = "http://localhost";
const applicationPort = "3000";
```

```
const backendAddress = "http://localhost";
const backendPort = "3001";
const secret = "cmpe281_nimbus_amazon";

//mongoDB
const connection_string =
  "mongodb+srv://sarthak:sarthak@amazonprototype-
vkmoj.mongodb.net/test?retryWrites=true&w=majority";

//mysql
const host = "http://amazonprototype.cxtlafrtsirj.us-east-2.rds.amazonaws.com/";
const password = "admin#123";

exports.applicationAddress = applicationAddress;
exports.applicationPort = applicationPort;
exports.backendAddress = backendAddress;
exports.backendPort = backendPort;
exports.secret = secret;
exports.connection_string = connection_string;
exports.host = host;
exports.password = password;
```

Mocha test

```
var app = require('../index');
var chai = require('chai');
chai.use(require('chai-http'));
var expect = require('chai').expect;

var agent = require('chai').request.agent(app);

describe('Amazon Prototype', function(){

  it('GET /product/admin/getProductCategory - Verifying category count',function(
done){
      agent.get('/product/admin/getProductCategory').send({})
          .then(function(res){
              // console.log(res.body);
              expect(res.body.length).to.equal(5);
              done();
          })
          .catch((e) => {
              done(e);
          });
  });

  it('GET /customer/orders/:email - Verifying order count',function(done){
    agent.get('/customer/orders/pranjal.jain@sjsu.edu').send({})
        .then(function(res){
```

```
            // console.log(res.body);
            expect(res.body.data.length).to.equal(2);
            done();
        })
        .catch((e) => {
            done(e);
        });
  });

  it('POST /customer/orders/list/cancel/product/:email - Verifying cancel order c
ount',function(done){
    agent.post('/customer/orders/list/cancel/product/pranjal.jain@sjsu.edu').send
({})
        .then(function(res){
            // console.log(res.body);
            expect(res.body.data.length).to.equal(1);
            done();
        })
        .catch((e) => {
            done(e);
        });
  });

  it('GET /customer/orders/list/open/product/:email - Verifying open order count'
,function(done){
    agent.get('/customer/orders/list/open/product/pranjal.jain@sjsu.edu').send({}
)
        .then(function(res){
            // console.log(res.body);
            expect(res.body.data.length).to.equal(0);
            done();
        })
        .catch((e) => {
            done(e);
        });
  });

  it('GET /customer/address/:id - Verifying address count',function(done){
    agent.get('/customer/address/5e955bb51616493fa824cdf0').send({})
        .then(function(res){
            // console.log(res.body);
            expect(res.body.addresses.length).to.equal(12);
            done();
        })
        .catch((e) => {
            done(e);
        });
  });

  it('GET /customer/payment/:id - Verifying payment card count',function(done){
    agent.get('/customer/payment/5e955bb51616493fa824cdf0').send({})
        .then(function(res){
```

```
            // console.log(res.body);
            expect(res.body.paymentCards.length).to.equal(7);
            done();
        })
        .catch((e) => {
            done(e);
        });
  });

  it('GET /customer/cartProducts/:id - Verifying saved product count',function(do
ne){
    agent.get('/customer/cartProducts/5e955bb51616493fa824cdf0').send({})
        .then(function(res){
            // console.log(res.body);
            expect(res.body.savedCnt).to.equal(14);
            done();
        })
        .catch((e) => {
            done(e);
        });
  });

  it('GET /customer/cartProducts/:id - Verifying cart product count',function(don
e){
    agent.get('/customer/cartProducts/5e955bb51616493fa824cdf0').send({})
        .then(function(res){
            // console.log(res.body);
            expect(res.body.cartCnt).to.equal(1);
            done();
        })
        .catch((e) => {
            done(e);
        });
  });

  it('GET admin/analytics/report6 - Verifying most viewed product count',function
(done){
    agent.get('/admin/analytics/report6').send({})
        .then(function(res){
            // console.log(res.body);
            expect(res.body.clicksArr.length).to.equal(7);
            done();
        })
        .catch((e) => {
            done(e);
        });
  });

  it('GET admin/analytics/report1 - Verifying number of orders per day count',fun
ction(done){
    agent.get('/admin/analytics/report1').send({})
        .then(function(res){
```

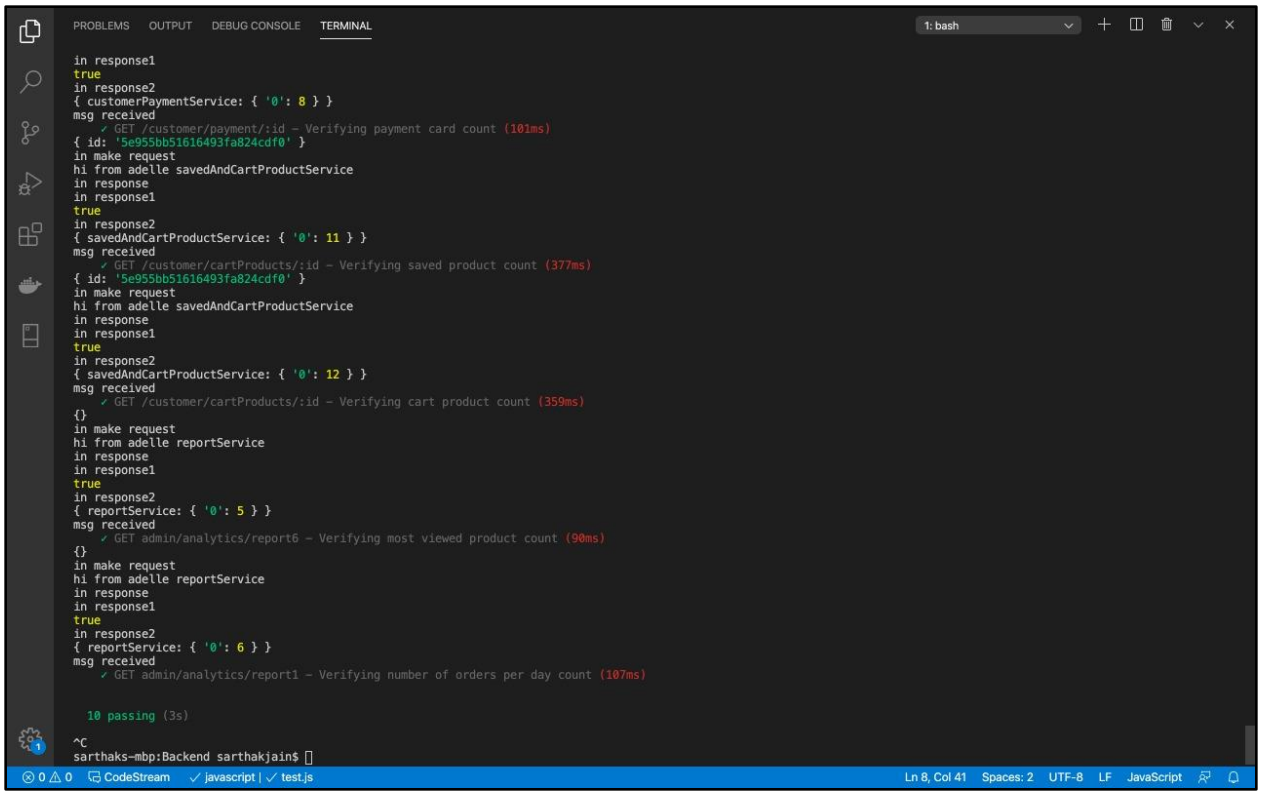
```
            // console.log(res.body);
            expect(res.body.adminReport1.length).to.equal(9);
            done();
        })
        .catch((e) => {
            done(e);
        });
  });

})
```

Results

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                   1: bash

sarthaks-mbp:Backend sarthakjain$ npm test

> backend@1.0.0 test /Users/sarthakjain/Desktop/AmazonPrototype/Backend
> mocha test

producer ready
Server listening on port 3001


  Amazon Prototype
Inside get of product/admin/getProductCategory
{}
in make request
hi from adelle adminProductService
1
returning next
in response
in response1
true
client ready!
in response2
{ adminProductService: { '0': 13 } }
msg received
    ✓ GET /product/admin/getProductCategory - Verifying category count (184ms)
Inside get of customer/orders/:emailId
{}
in make request
hi from adelle orderAddressService
in response
in response1
true
in response2
{ orderAddressService: { '0': 36 } }
msg received
{ data:
   [ { _id: 'b2c032fd-0cd3-419f-b1d7-32b81e00fb78',
       Product_id: '5ea498742c73295362bcf477',
       quantity: 1,
       TotalPrice: 502,
       Status: '6',
       seller_email_id: '2013uec1108@mnit.ac.in',
       customer_email_id: 'pranjal.jain@sjsu.edu',
       cancelProduct: false,
       giftFlag: false,
       giftmsg: '',
       createdAt: '2020-05-07T23:03:49.000Z',
       updatedAt: '2020-05-07T23:04:56.000Z',
       order_id: '93733056-34b9-403e-bd6c-1096e8e59146',
       Order: [Object],
       products: [Object] },
     { _id: 'a4f2eca0-386c-4e2d-815a-29d603258a6d',
       Product_id: '5eb44f06995f3778fcccb16e',
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                   1: bash

while returning
    ✓ GET /customer/orders/:email - Verifying order count (769ms)
Inside post of customer/orders/list/cancel/product/:emailId
{}
in make request
hi from adelle orderAddressService
in response
in response1
true
in response2
{ orderAddressService: { '0': 37 } }
msg received
{ data:
   [ { _id: 'f53517ea-1742-4a7d-b56d-ea442fd06752',
       Product_id: '5e9e586d16ee40493ff37068',
       quantity: 1,
       TotalPrice: 352.34,
       Status: '3',
       seller_email_id: '2013uec1108@mnit.ac.in',
       customer_email_id: 'pranjal.jain@sjsu.edu',
       cancelProduct: true,
       giftFlag: false,
       giftmsg: '',
       createdAt: '2020-05-07T20:49:44.000Z',
       updatedAt: '2020-05-07T20:52:07.000Z',
       order_id: '2bf5c2ec-fb92-4e91-9596-18df3f383b13',
       Order: [Object],
       products: [Object] } ],
  status: 200 }
while returning
    ✓ POST /customer/orders/list/cancel/product/:email - Verifying cancel order count (220ms)
Inside get of customer/orders/list/open/product/:emailId
{}
in make request
hi from adelle orderAddressService
in response
in response1
true
in response2
{ orderAddressService: { '0': 38 } }
msg received
{ data: [], status: 200 }
while returning
    ✓ GET /customer/orders/list/open/product/:email - Verifying open order count (138ms)
{}
in make request
hi from adelle customerAddressService
in response
in response1
true
in response2
{ customerAddressService: { '0': 10 } }
⊗ 0 ⚠ 0   CodeStream   ✓ javascript | ✓ test.js                    Ln 8, Col 41   Spaces: 2   UTF-8   LF   JavaScript
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                   1: bash

msg received
    ✓ GET /customer/address/:id - Verifying address count (130ms)
{}
in make request
hi from adelle customerPaymentService
in response
in response1
true
in response2
{ customerPaymentService: { '0': 8 } }
msg received
    ✓ GET /customer/payment/:id - Verifying payment card count (101ms)
{ id: '5e955bb51616493fa824cdf0' }
in make request
hi from adelle savedAndCartProductService
in response
in response1
true
in response2
{ savedAndCartProductService: { '0': 11 } }
msg received
    ✓ GET /customer/cartProducts/:id - Verifying saved product count (377ms)
{ id: '5e955bb51616493fa824cdf0' }
in make request
hi from adelle savedAndCartProductService
in response
in response1
true
in response2
{ savedAndCartProductService: { '0': 12 } }
msg received
    ✓ GET /customer/cartProducts/:id - Verifying cart product count (359ms)
{}
in make request
hi from adelle reportService
in response
in response1
true
in response2
{ reportService: { '0': 5 } }
msg received
    ✓ GET admin/analytics/report6 - Verifying most viewed product count (90ms)
{}
in make request
hi from adelle reportService
in response
in response1
true
in response2
{ reportService: { '0': 6 } }
msg received
    ✓ GET admin/analytics/report1 - Verifying number of orders per day count (107ms)
⊗ 0 ⚠ 0   CodeStream   ✓ javascript | ✓ test.js                    Ln 8, Col 41   Spaces: 2   UTF-8   LF   JavaScript
```

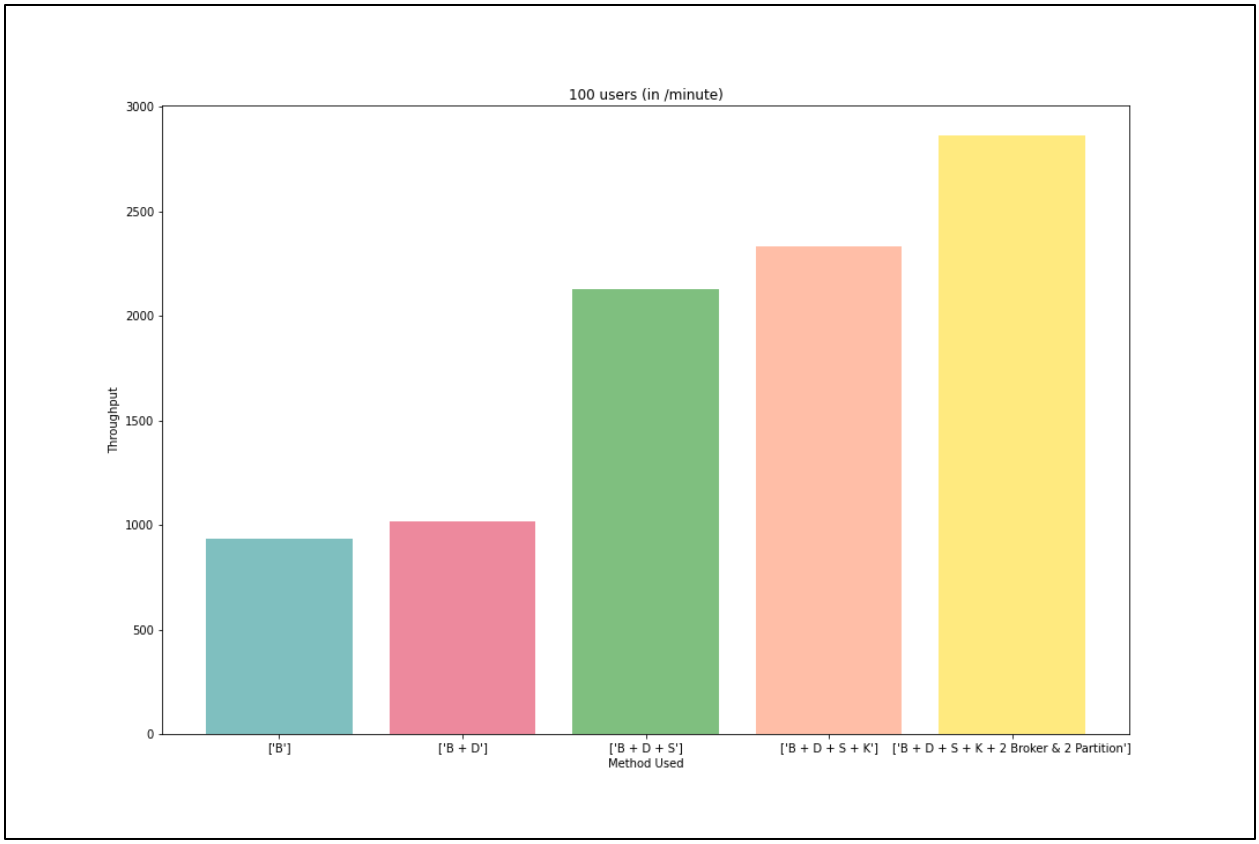


Application Performance

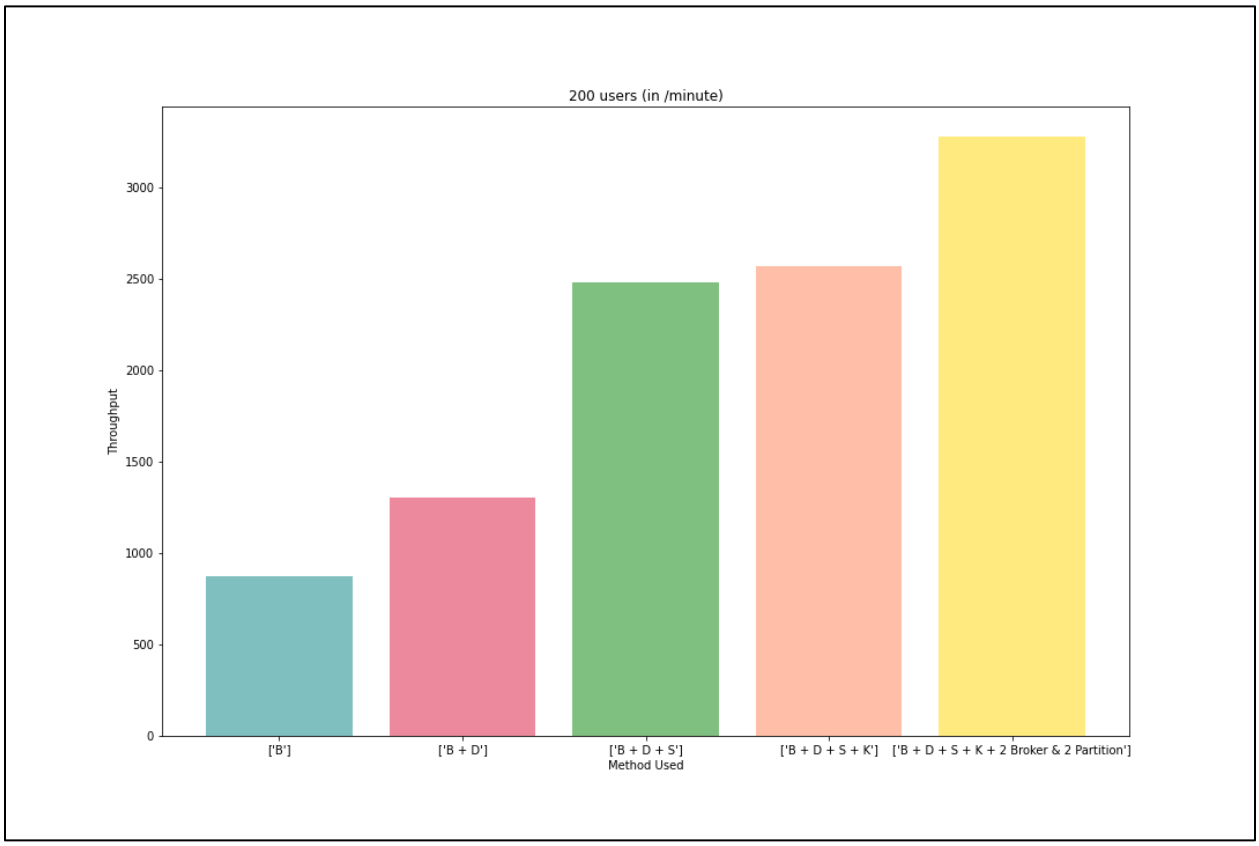
	100 users (in /minute)	200 users (in /minute)	300 users (in /minute)	400 users (in /minute)	500 users (in /minute)
B	932	872	847	1,280	1,344
B + D	1,017	1,303	1,379	1,324	1,358
B + D + S	2,127	2,482	2,782	3,049	3,199
B + D + S + K	2,333	2,569	2,886	3,153	3,213
B + D + S + K + 2 Broker & 2 Partition	2,862	3,277	3,856	4,227	4,672

Throughput values in per minute

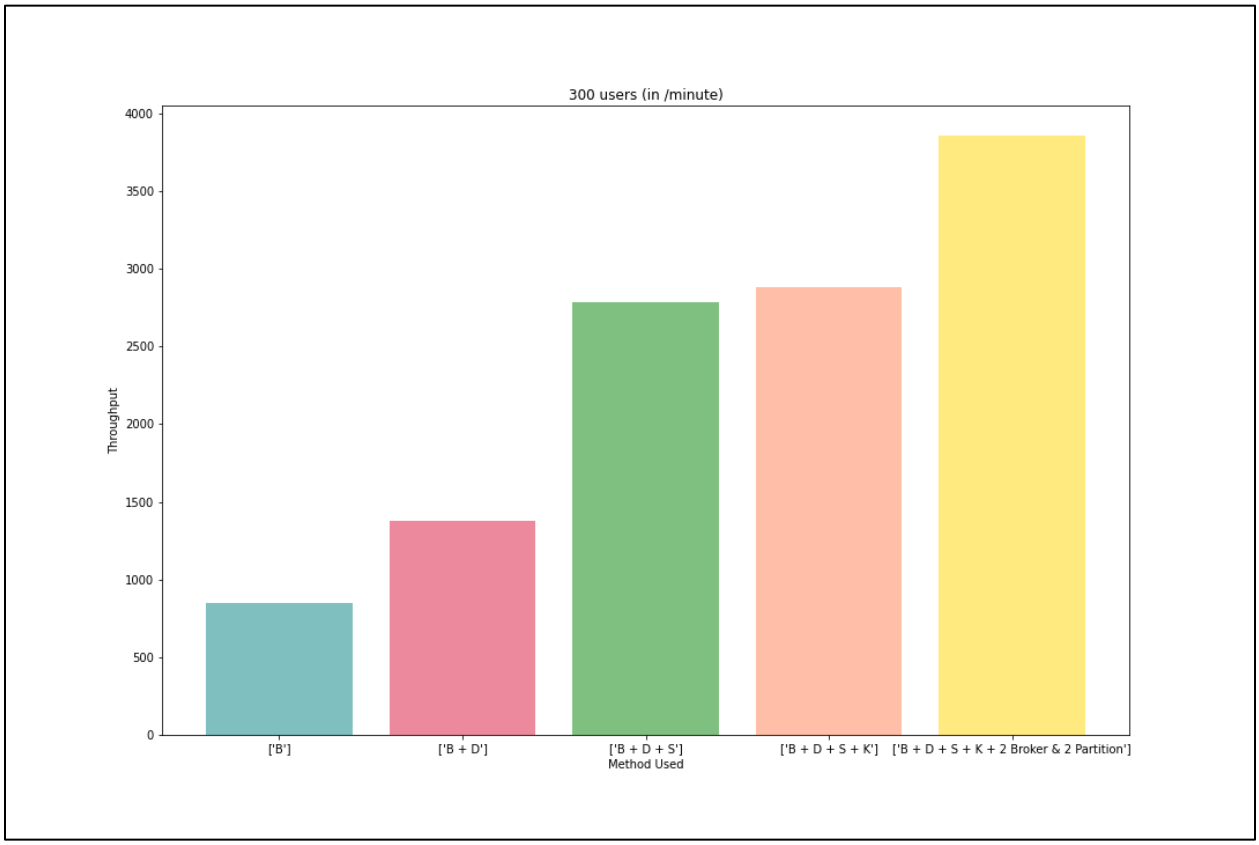
100 users


100 users (in /minute)
Throughput
3000
2500
2000
1500
1000
500
0
['B']
['B + D']
['B + D + S']
['B + D + S + K']
['B + D + S + K + 2 Broker & 2 Partition']
Method Used

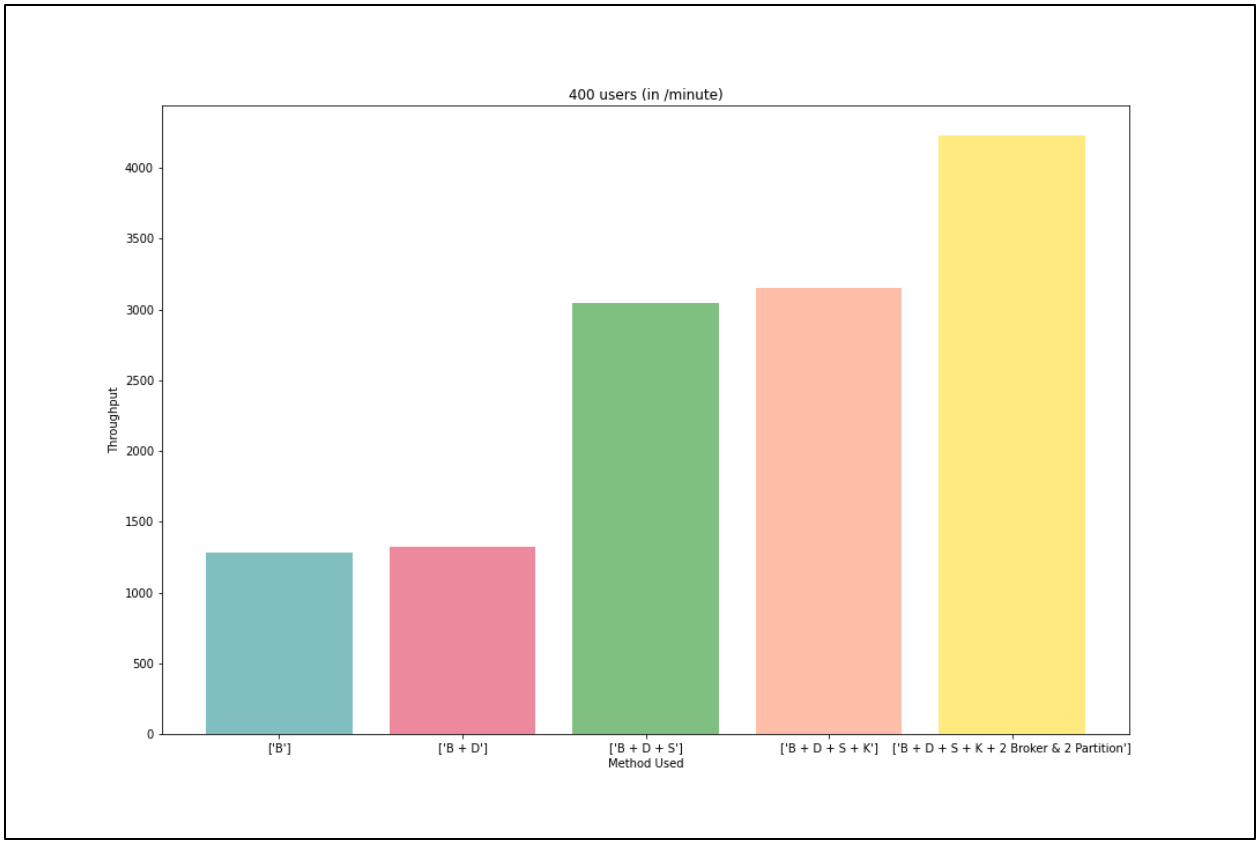
200 users


200 users (in /minute)
Throughput
3000
2500
2000
1500
1000
500
0
['B']
['B + D']
['B + D + S']
['B + D + S + K']
['B + D + S + K + 2 Broker & 2 Partition']
Method Used

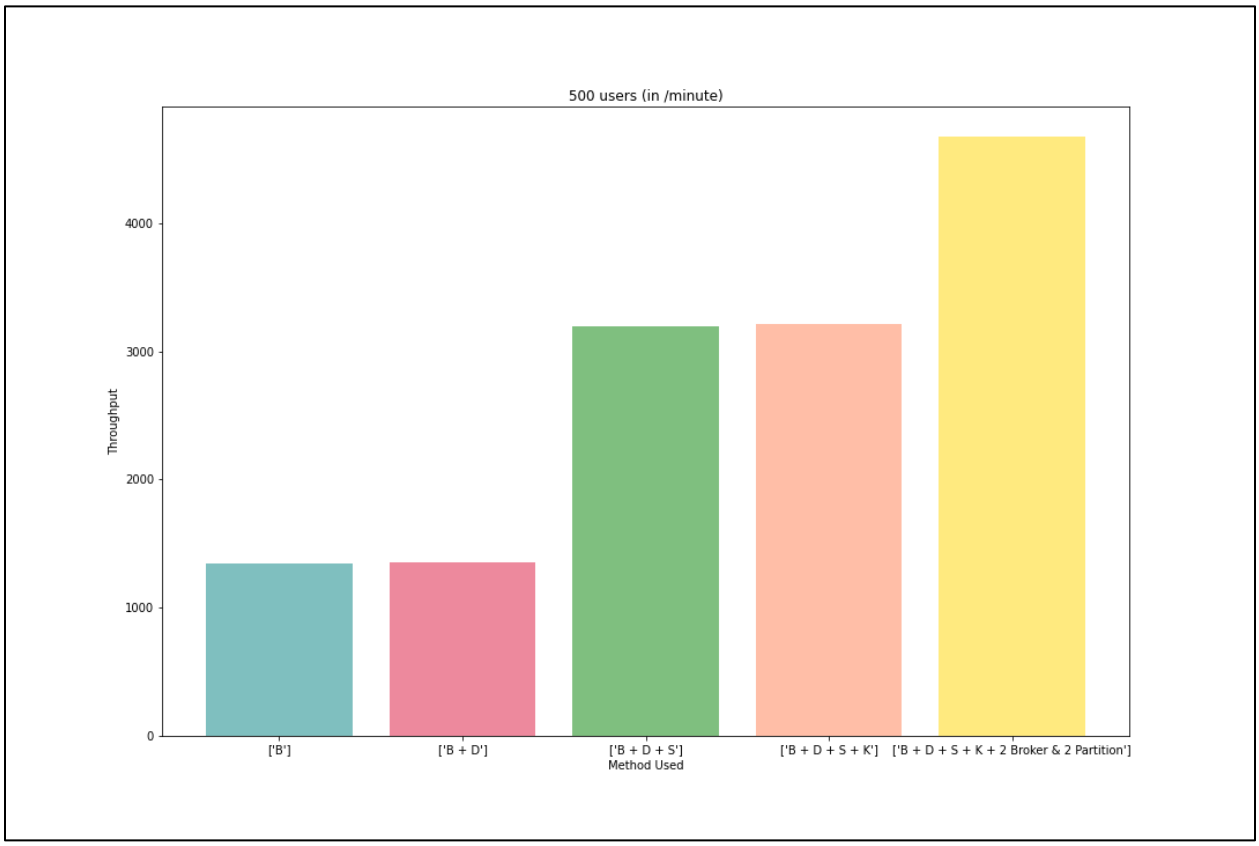
300 users



400 users


400 users (in /minute)
Throughput
0
500
1000
1500
2000
2500
3000
3500
4000
['B']
['B + D']
['B + D + S']
['B + D + S + K']
['B + D + S + K + 2 Broker & 2 Partition']
Method Used

500 users


500 users (in /minute)
Throughput
0
1000
2000
3000
4000
['B']
['B + D']
['B + D + S']
['B + D + S + K']
['B + D + S + K + 2 Broker & 2 Partition']
Method Used