Microservice Backend Application

The application is a backend system built using a microservices architecture, designed to handle various functionalities related to transactions, orders, users, and products. It is primarily targeted at managing the backend operations of an e-commerce platform or similar service.

Key Components:

Microservices Structure: The application is divided into separate microservices, each responsible for handling specific tasks. These microservices include Transaction Service, Order Service, User Service, and Product Service.

Functionality: Each microservice encapsulates the logic and data related to its respective domain. For example, the Order Service manages orders, including tracking numbers, products ordered, customer information, and quantities.

Data Management: MongoDB is used as the database management system for storing and retrieving data efficiently. Each microservice has its own set of models, controllers, and routes to interact with the database.

Asynchronous Communication: Apache Kafka is employed for asynchronous communication between microservices. It serves as a distributed event streaming platform, facilitating real-time data pipelines and streaming applications.

Web Framework: Fastify, a fast and low overhead web framework for Node.js, is utilized for building HTTP APIs. It ensures high performance and efficiency in handling incoming requests and responses.

Real-time Communication: Socket.io is integrated to enable real-time communication between clients and servers. This is particularly useful for implementing features like live updates, notifications, and chat functionalities.

Summary:

Overall, the application provides a scalable, efficient, and modular backend infrastructure for managing various aspects of an e-commerce platform or similar service. It leverages microservices architecture, MongoDB for data storage, Apache Kafka for asynchronous communication, Fastify for web API development, and Socket.io for real-time communication, ensuring robustness, flexibility, and responsiveness in handling diverse business requirements.

Project Structure

```
/backend_assignment_node_microservices
|   README.md
|   package.json
|
├──transaction-service
|   ├──models
|   |   transaction.js
|   ├──controllers
|   |   transactionController.js
|   └──routes
|       transactionRoutes.js
|
├──order-service
|   ├──models
|   |   order.js
|   ├──controllers
|   |   orderController.js
|   └──routes
|       orderRoutes.js
|
├──user-service
|   ├──models
|   |   user.js
|   ├──controllers
|   |   userController.js
|   └──routes
|      userRoutes.js
|
├──product-service
|   ├──models
|   |   product.js
|   ├──controllers
|   |   productController.js
|   └──routes
|      productRoutes.js
|
└──kafka
    |  producer.js
    |  consumer.js
```

OrderServiceSchema.js

```
const mongoose = require('mongoose');
const orderSchema = new mongoose.Schema({
   transactionNumber: String,
   trackingNumber: String,
   productOrdered: String,
   productDescription: String,
   qty: String,
   fullName: String,
   address: String,
   city: String,
   state: String,
   telephone: String
});
const Order = mongoose.model('Order', orderSchema);
module.exports = Order;
```

Required Tools:

Node.js: A JavaScript runtime for building scalable network applications.

MongoDB: A NoSQL database used for storing and retrieving data efficiently.

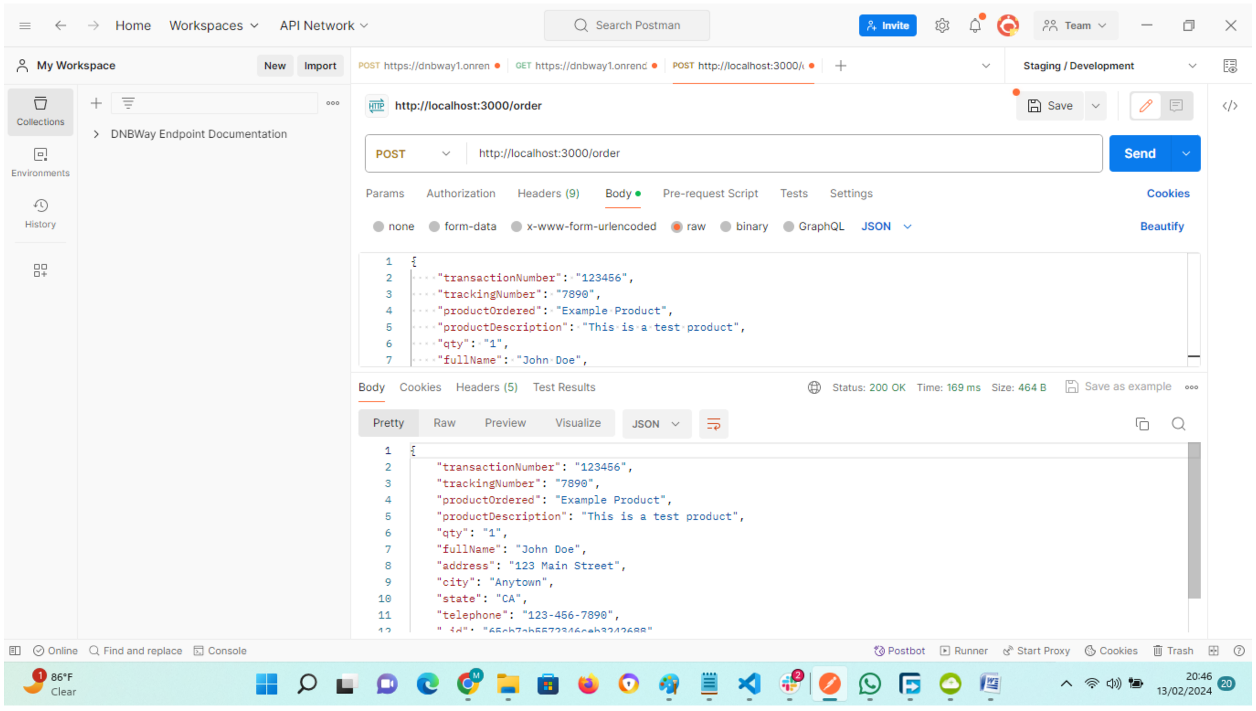
Apache Kafka: A distributed event streaming platform used for building real-time data pipelines and streaming applications.

Fastify: A fast and low overhead web framework for Node.js.

Github Repo

Codebase can be found here
https://github.com/EmekaIwuagwu/Backend_Assignment_NodeMicroservices

Screenshots of REST API



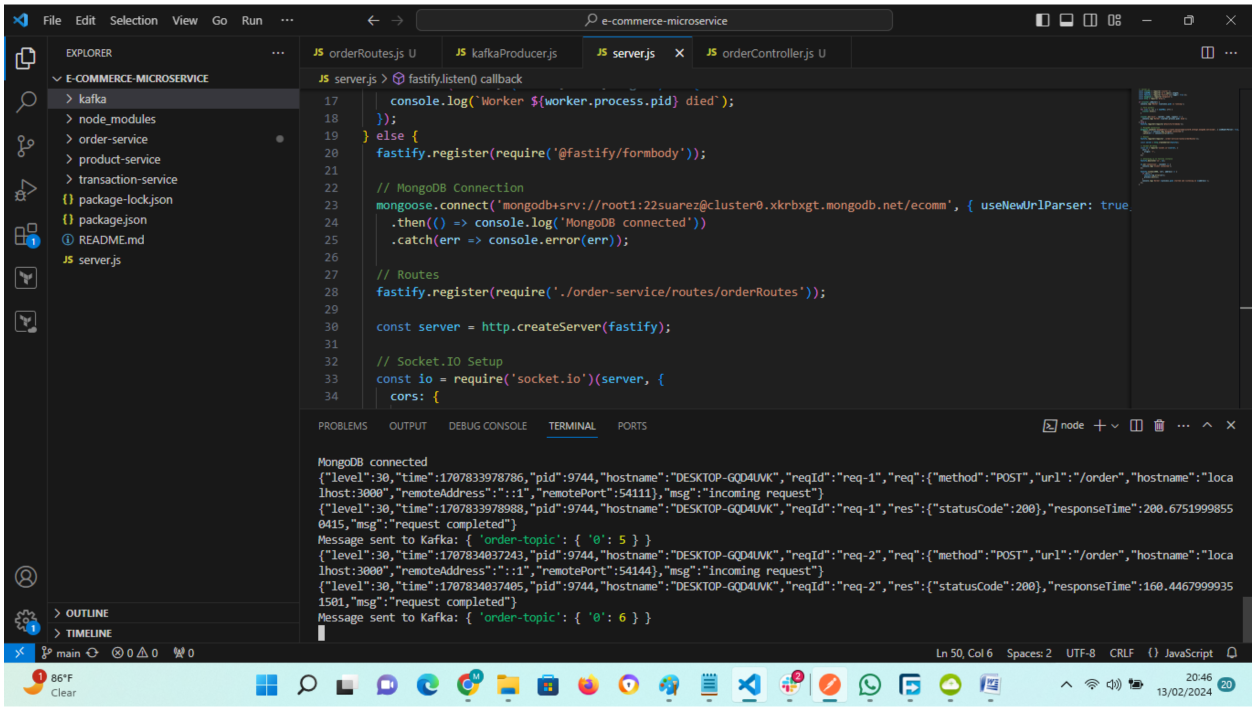
http://localhost:3000/order

POST http://localhost:3000/order

```
{
    "transactionNumber": "123456",
    "trackingNumber": "7890",
    "productOrdered": "Example Product",
    "productDescription": "This is a test product",
    "qty": "1",
    "fullName": "John Doe",
```

Status: 200 OK Time: 169 ms Size: 464 B

```
{
    "transactionNumber": "123456",
    "trackingNumber": "7890",
    "productOrdered": "Example Product",
    "productDescription": "This is a test product",
    "qty": "1",
    "fullName": "John Doe",
    "address": "123 Main Street",
    "city": "Anytown",
    "state": "CA",
    "telephone": "123-456-7890",
```



```
    console.log(`Worker ${worker.process.pid} died`);
  });
} else {
  fastify.register(require('@fastify/formbody'));

  // MongoDB Connection
  mongoose.connect('mongodb+srv://root1:22suarez@cluster0.xkrbxgt.mongodb.net/ecomm', { useNewUrlParser: true
    .then(() => console.log('MongoDB connected'))
    .catch(err => console.error(err));

  // Routes
  fastify.register(require('./order-service/routes/orderRoutes'));

  const server = http.createServer(fastify);

  // Socket.IO Setup
  const io = require('socket.io')(server, {
    cors: {
```

```
MongoDB connected
{"level":30,"time":1707833978786,"pid":9744,"hostname":"DESKTOP-GQD4UVK","reqId":"req-1","req":{"method":"POST","url":"/order","hostname":"loca
lhost:3000","remoteAddress":"::1","remotePort":54111},"msg":"incoming request"}
{"level":30,"time":1707833978988,"pid":9744,"hostname":"DESKTOP-GQD4UVK","reqId":"req-1","res":{"statusCode":200},"responseTime":200.6751999855
0415,"msg":"request completed"}
Message sent to Kafka: { 'order-topic': { '0': 5 } }
{"level":30,"time":1707834037243,"pid":9744,"hostname":"DESKTOP-GQD4UVK","reqId":"req-2","req":{"method":"POST","url":"/order","hostname":"loca
lhost:3000","remoteAddress":"::1","remotePort":54144},"msg":"incoming request"}
{"level":30,"time":1707834037405,"pid":9744,"hostname":"DESKTOP-GQD4UVK","reqId":"req-2","res":{"statusCode":200},"responseTime":160.4467999935
1501,"msg":"request completed"}
Message sent to Kafka: { 'order-topic': { '0': 6 } }
```