Assignment-5
CS 461

Name : Vishal Kumar Anand
ID:201951170
CSE

Problem: Consider that two applications (Let X and Y) are running on two different locations. Let's say X is directly communicating with customers (Like any E-commerce) and Y is using the information related to customers. Application Y wants to get a list of all the customers of application X and also analyze the communication time between machine X and Y.

(NOTE: You should have the database at the location of application X which contains the database table Customer(CutomerID, Customer First Name, Customer Last Name, Customer Email, Customer Mobile). If you don't have the database at the location of application X then you can create a text file containing CutomerID, Customer First Name, Customer Last Name, Customer Email, Customer Mobile, and Customer balance. Fill in the random entries in the text file for now.).

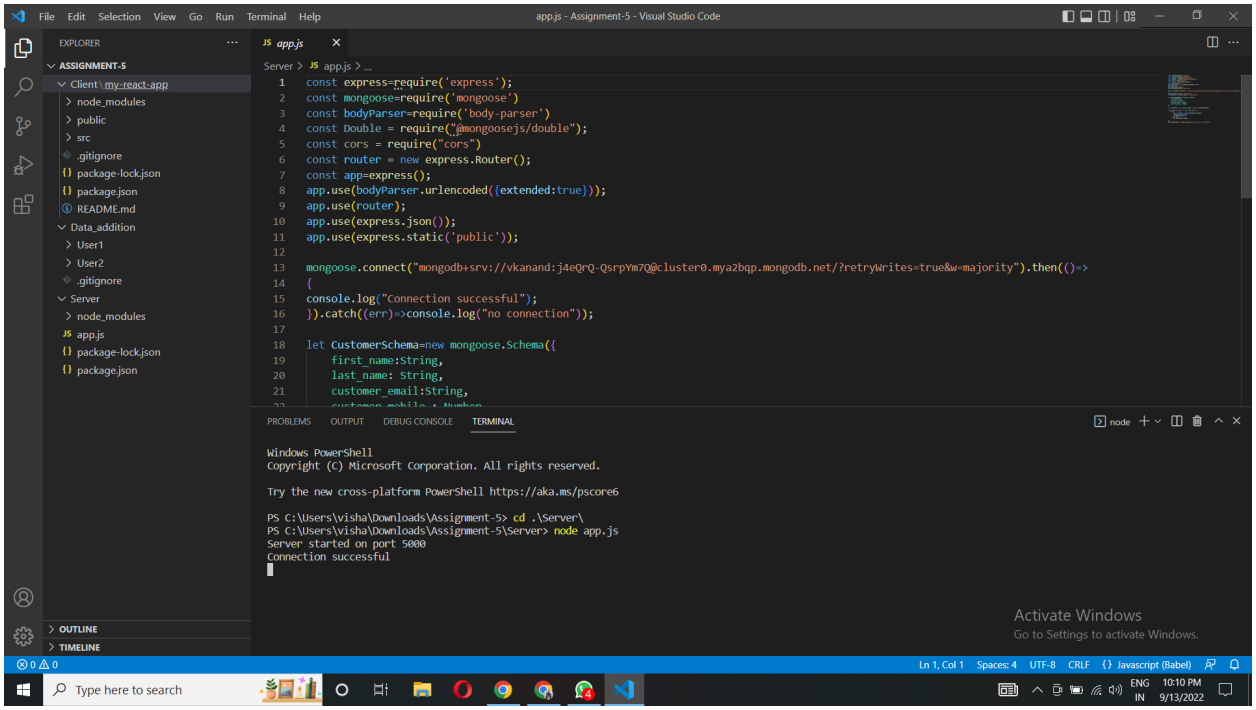
Server code(API) : {X}

Connecting MongoDB database with the server using the cluster URL from MongoDB Atlas…

```
const express=require('express');
const mongoose=require('mongoose')
const bodyParser=require('body-parser')
const Double = require("@mongoosejs/double");
const cors = require("cors")
const router = new express.Router();
const app=express();
app.use(bodyParser.urlencoded({extended:true}));
app.use(router);
```

```
app.use(express.json());
app.use(express.static('public'));

mongoose.connect("mongodb+srv://vkanand:j4eQrQ-QsrpYm7Q@cluster0.mya2bqp.mongodb.net/?retryWrites=true&w=majority").then(()=>
{
console.log("Connection successful");
}).catch((err)=>console.log("no connection"));

let CustomerSchema=new mongoose.Schema({
    first_name:String,
    last_name: String,
    customer_email:String,
    customer_mobile : Number,
    customer_balance : Number
});

let Customerdt = new mongoose.model('customer',CustomerSchema);

router.get("/customers", async (req, res) => {
    try {
        const customers = await Customerdt.find({});
        console.log(customers);
        res.send(customers);
    } catch (e) {
        res.status(500).send();
    }
});
app.listen(5000,()=>console.log("Server started on port 5000"));
```

Screenshot :



Code for adding data in database by making form and then accessing it:-

Created form to take input from user to add data in mongoDB database …

create.ejs :

```
<div>
  <form class="customer" action="/" method="post">
    <input
      type="number"
      name="customer_balance"
      placeholder="Customer Balance"
      autocomplete="off"
    />
    <input
      type="text"
      name="customer_email"
      placeholder="Customer Email"
      autocomplete="off"
```

```
    />
    <input
      type="number"
      name="customer_mobile"
      placeholder="Customer Mobile"
      autocomplete="off"
    />
    <input
      type="text"
      name="first_name"
      placeholder="Customer First Name"
      autocomplete="off"
    />
    <input
      type="text"
      name="last_name"
      placeholder="Customer Last Name"
      autocomplete="off"
    />
    <button type="submit" name="Submit">Submit</button>
  </form>
</div>
```

Server1.js :

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();

app.set('view engine', 'ejs');

app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static('public'));
```

```
mongoose.connect(

'mongodb+srv://newuser:Neeraj27@cluster0.mfuj1tc.mongodb.net/?retryWrites=
true&w=majority',

  {
    useNewUrlParser: true,
  }
);

const customerSchema = new mongoose.Schema({
  // Customer_ID: String,
  customer_balance: Number,
  customer_email: String,
  customer_mobile: Number,
  first_name: String,
  last_name: String,
});

const Customer = mongoose.model('Customer', customerSchema);

app.get('/', function (req, res) {
  // Customer.find({}, function (err, results) {
  //   if (results.length === 0) {
  //     Customer.insertMany(dummyCustomer, function (err) {
  //       if (err) {
  //         console.log(err);
  //       } else {
  //         console.log('Successfully added dummyCustomer in DataBase');
  //       }
  //     });
  //     res.render('create');
  //   } else {
  res.render('create');
  // }
  // });
});

app.post('/', function (req, res) {
```

```
  //redirect to different routes after post request from here rather than
making new post requests app.post

  const Customers_first_name = req.body.first_name;
  const Customers_last_name = req.body.last_name;
  const Customers_Mobile = req.body.customer_mobile;
  const Customers_Email = req.body.customer_email;
  const Customer_Balance = req.body.customer_balance;

  const addedCustomer = new Customer({

    first_name: Customers_first_name,
    last_name: Customers_last_name,
    customer_mobile: Customers_Mobile,
    customer_email: Customers_Email,
    customer_balance: Customer_Balance,
  });

  addedCustomer.save();
  res.redirect('/');
});

app.listen(3001, function () {
  console.log('Server started successfully at port:3001!');
});
```

Client side code: {Y}

For fetching data and displaying…

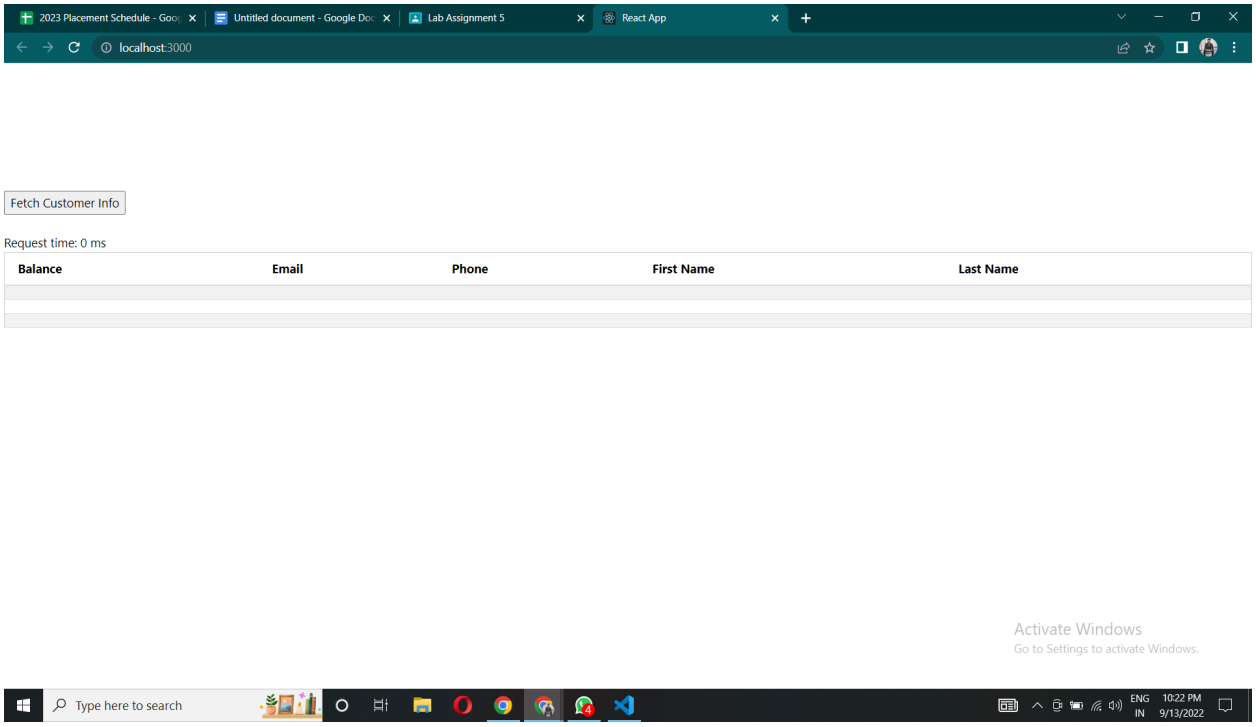
App.js :-

```
import Axios from "axios"
import { useState } from 'react';

function App() {
```

```
  const [customerData, setCustomerData] = useState(["fddfdf", "fdfdff",
"dfdfdd"]);
  const [time, setTime] = useState(0);
  const getUserData = async () => {
    try {
      var start = new Date().getTime();
      let customerDataResp = await
Axios.get('http://localhost:5000/customers');
      var end = new Date().getTime();
      setTime(end - start);
      console.log(customerDataResp)
      setCustomerData(customerDataResp.data);
    } catch (err) {
      console.log(err.message)
    }
  }

  return (
    <div className="App">
      <button style={{ marginTop: "10%" }} onClick={() =>
getUserData()}>Fetch Customer Info</button> <br /> <br />
      Request time: {time} ms
      <table class="w3-table-all">
        <tr>
          <th>
            Balance
          </th>
          <th>
            Email
          </th>
          <th>
            Phone
          </th>
          <th>
            First Name
          </th>
          <th>
            Last Name
          </th>
        </tr>
```

```
        {customerData.map((ele, index) => <tr>
          <td>
            {ele.customer_balance}
          </td>
          <td>
          {ele.customer_mobile}
        </td>
          <td>
            {ele.customer_email}
          </td>
          <td>
            {ele.first_name}
          </td>
          <td>
            {ele.last_name}
          </td>
        </tr>)}
      </table>
    </div>
  );
}

export default App;
```

Screenshot of client display before fetching info :



Screenshot of client display after fetching info :

Fetch Customer Info

Request time: 86 ms

Balance	Email	Mobile	First Name	Last Name
4552	vkanand2207@gmail.com	1234567890	vk	anand
95122	vishalanand@gmail.com	8090167364	Vishal Kumar	Anand

Communication time between Y client and server X for a single request to be 86ms. As in the first time flag at the start of making the request is stored and then again time flag is marked after the request is completed and calculating the difference of both time flags.