Setting up all the MongoDB files to practice on your laptop: (ONLY if you want to try this in your Laptop else just skip this part and directly go to Experiment 1)

Links:

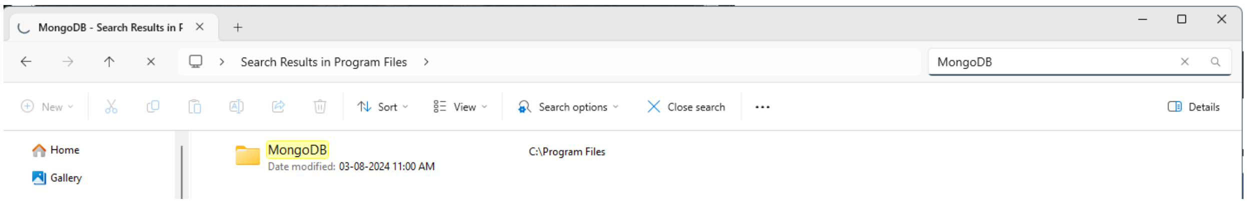
Mongosh: https://downloads.mongodb.com/compass/mongosh-2.2.15-x64.msi

Mongo Database Tools: https://fastdl.mongodb.org/tools/db/mongodb-database-tools-windows-x86_64-100.10.0.msi

Mongo Compass: https://downloads.mongodb.com/compass/mongodb-compass-1.43.5-win32-x64.exe

Install them both and restart the computer

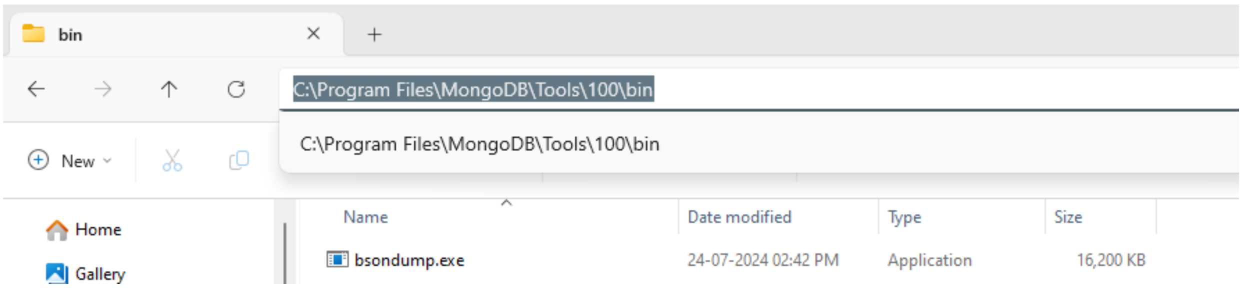
Now open C drive and in the search bar type 'MongoDB'



MongoDB
Date modified: 03-08-2024 11:00 AM
C:\Program Files

From there go to MongoDB > Tools > 100 > bin

Copy as Path in the Address bar

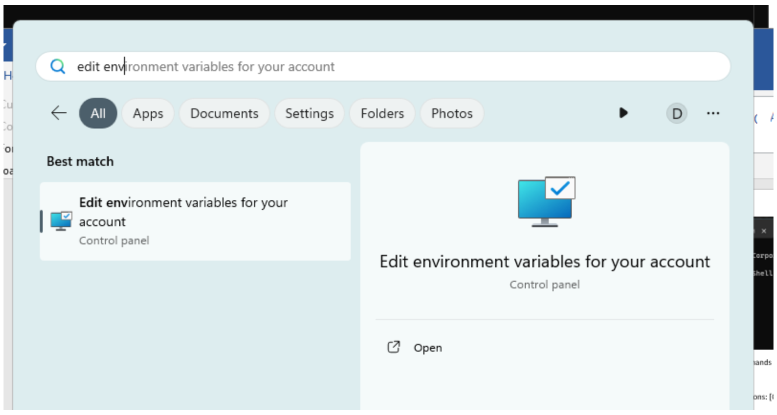


C:\Program Files\MongoDB\Tools\100\bin

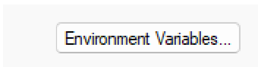
C:\Program Files\MongoDB\Tools\100\bin

Name	Date modified	Type	Size
bsondump.exe	24-07-2024 02:42 PM	Application	16,200 KB

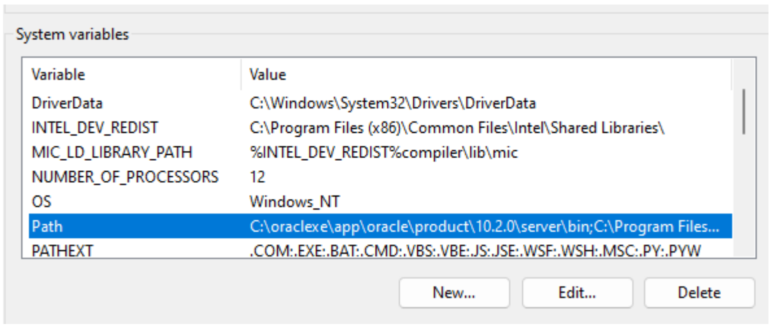
In the start menu search bar type 'Edit environment'


edit environment variables for your account
All
Apps
Documents
Settings
Folders
Photos
Best match
Edit environment variables for your account
Control panel
Edit environment variables for your account
Control panel
Open

In the Window Click 'Environment Variables

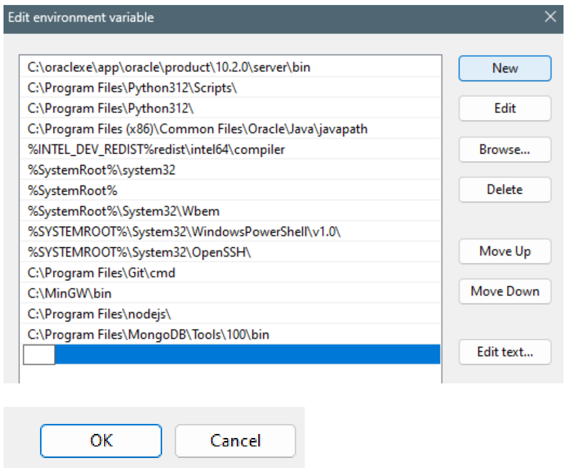


In the system variables window click Path and click the 'Edit; Button



Variable	Value
DriverData	C:\Windows\System32\Drivers\DriverData
INTEL_DEV_REDIST	C:\Program Files (x86)\Common Files\Intel\Shared Libraries\
MIC_LD_LIBRARY_PATH	%INTEL_DEV_REDIST%compiler\lib\mic
NUMBER_OF_PROCESSORS	12
OS	Windows_NT
Path	C:\oraclexe\app\oracle\product\10.2.0\server\bin;C:\Program Files...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW

Click New and the Box paste you address and hit Ok > in all the previous three windows

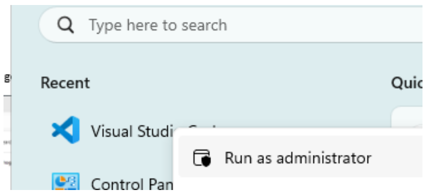


```
C:\oraclexe\app\oracle\product\10.2.0\server\bin
C:\Program Files\Python312\Scripts\
C:\Program Files\Python312\
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%INTEL_DEV_REDIST%redist\intel64\compiler
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\MinGW\bin
C:\Program Files\nodejs\
C:\Program Files\MongoDB\Tools\100\bin
```

(This was all just for the 1st program)(Yes this Subject is a Joke)

(Hopefully this is all preinstalled in the Lab exam… these steps are only if YOU want to try it in your Laptop)

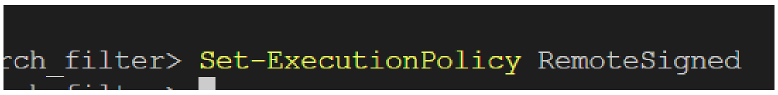
Note: If any problem occurs while running npm commands then



Run VS code as admisistrator

Then in the terminal run the command

```
Set-ExecutionPolicy RemoteSigned
```



```
ch_filter> Set-ExecutionPolicy RemoteSigned
```

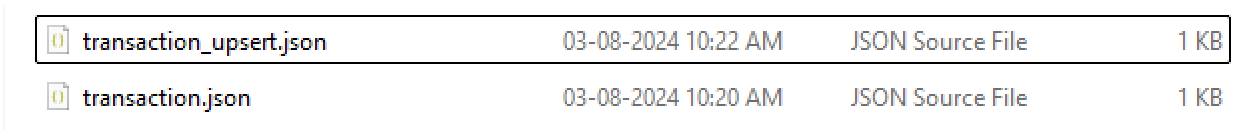
1 a. Using MongoDB, create a collection called transactions in database usermanaged (drop if it already exists)

and bulk load the data from a json file, transactions.json

b. Upsert the record from the new file called transactions_upsert.json in Mongodb.

Step 1:

Create Two Files transaction.json and transaction_upsert.json



transaction_upsert.json	03-08-2024 10:22 AM	JSON Source File	1 KB
transaction.json	03-08-2024 10:20 AM	JSON Source File	1 KB

Open These two files in VS code and put the values

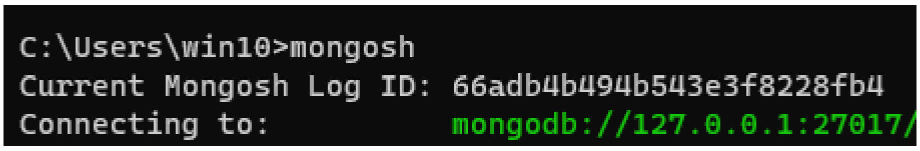
1.Transaction.json

```json
[
    {
      "ID": "1",
      "Name": "Somu",
      "payment": {
        "Total": 150.75
      },
      "transaction": {
        "Price": 150.75
      }
    },
    {
      "ID": "2",
      "Name": "Ravi",
      "payment": {
        "Total": 200.00
      },
      "transaction": {
        "Price": 200.00
      }
    },
    {
      "ID": "3",
      "Name": "Somu",
      "payment": {
        "Total": 320.50
      },
      "transaction": {
        "Price": 320.50
      }
    }
  ]
```

2.transaction_upsert.json

```json
[
    {
      "ID": "3",
      "Name": "Anusha",
      "payment": {
        "Total": 75.25
      },
      "transaction": {
        "Price": 75.25
      }
    },
    {
      "ID": "4",
      "Name": "Bhuvan",
      "payment": {
        "Total": 500.00
      },
      "transaction": {
        "Price": 500.00
      }
    }
  ]
```

Step 2: Open Command Prompt and type ‘mongosh’

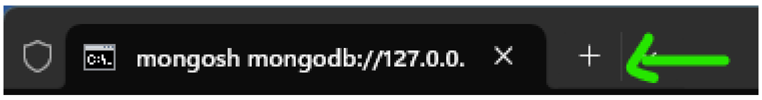


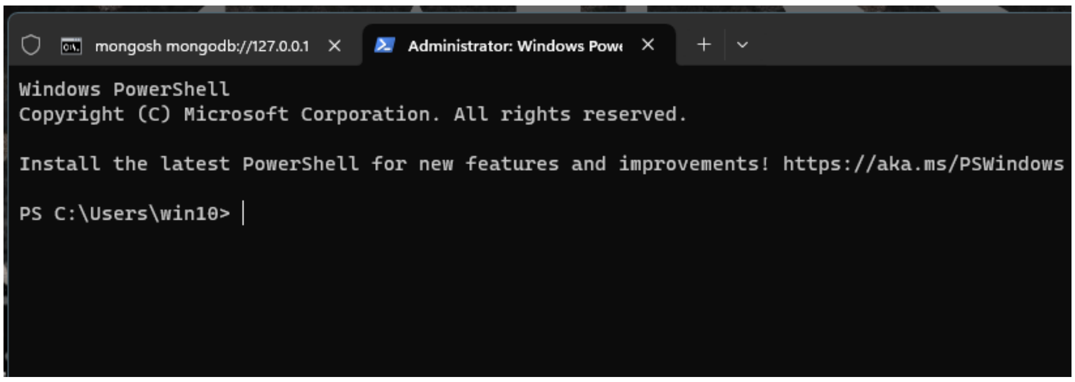
```
C:\Users\win10>mongosh
Current Mongosh Log ID: 66adb4b494b543e3f8228fb4
Connecting to:          mongodb://127.0.0.1:27017/
```

Step 3: Run the following commands

```
use myDatabase;

db.transactions.drop() //If it already exists

db.createCollection("transactions");
```

Step 4: Click the + sign on the title bar or just open a NEW command prompt window





```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\win10> |
```

Step 5: Run the following commands in the NEW Command Prompt

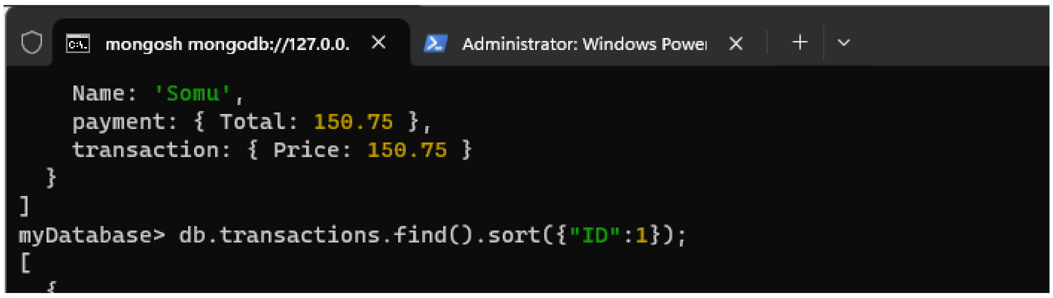
Copy the path of your Transaction.json File (Change the address to the transaction.json instead of "C:\Users\win10\Desktop\mern\transaction.json" put your address. The command should come in one whole line)

```
mongoimport --db myDatabase --collection transactions –file
C:\Users\win10\Desktop\mern\transaction.json --jsonArray
```

MAKE Sure to change the Path to your File's Path

Step 6: In the MongoDB terminal Type

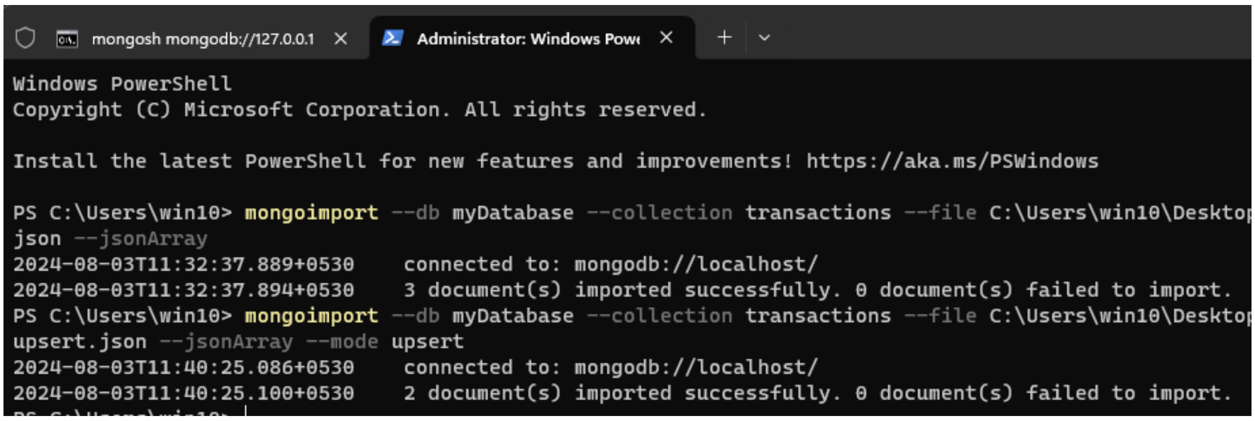
```
db.transactions.find().sort({"ID":1});
```



```
    Name: 'Somu',
    payment: { Total: 150.75 },
    transaction: { Price: 150.75 }
  }
]
myDatabase> db.transactions.find().sort({"ID":1});
[
```

Step 7: Again in the 2nd Command Prompt type the command (Change the address to the transaction_upsert.json instead of "C:\Users\win10\Desktop\mern\transaction_upsert.json" put your address. The command should come in one whole line)

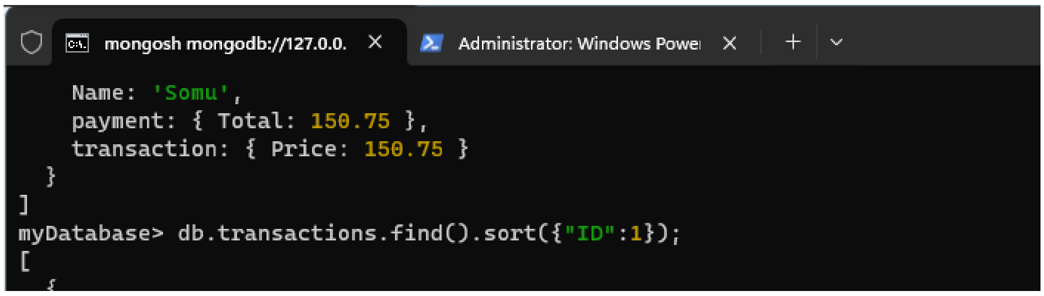
```
mongoimport --db myDatabase --collection transactions –file
C:\Users\win10\Desktop\mern\transaction_upsert.json --jsonArray --mode upsert
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\win10> mongoimport --db myDatabase --collection transactions --file C:\Users\win10\Deskto
json --jsonArray
2024-08-03T11:32:37.889+0530    connected to: mongodb://localhost/
2024-08-03T11:32:37.894+0530    3 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\win10> mongoimport --db myDatabase --collection transactions --file C:\Users\win10\Deskto
upsert.json --jsonArray --mode upsert
2024-08-03T11:40:25.086+0530    connected to: mongodb://localhost/
2024-08-03T11:40:25.100+0530    2 document(s) imported successfully. 0 document(s) failed to import.
```

Step 8: In the MongoDB terminal Type

```
db.transactions.find().sort({"ID":1});
```



```
    Name: 'Somu',
    payment: { Total: 150.75 },
    transaction: { Price: 150.75 }
  }
]
myDatabase> db.transactions.find().sort({"ID":1});
[
```

2. Query MongoDB with Conditions: [Create appropriate collection with necessary documents to answer the query]

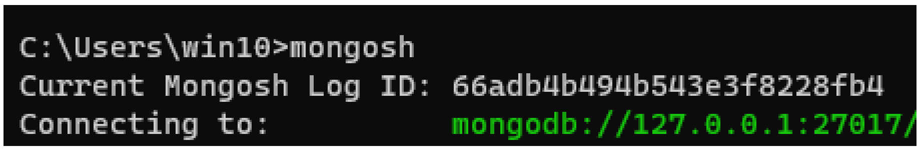
a. Find any record where Name is Somu

b. Find any record where total payment amount (Payment.Total) is 600.

c. Find any record where price (Transaction.price) is between 300 to 500.

d. Calculate the total transaction amount by adding up Payment.Total in all records.

Step 1: Open Command Prompt and type 'mongosh'



```
C:\Users\win10>mongosh
Current Mongosh Log ID: 66adb4b494b543e3f8228fb4
Connecting to:          mongodb://127.0.0.1:27017/
```

Step 2: Run the following commands

```
use myDatabase;

db.createCollection("transactions");

db.transactions.insertOne({"ID":"1","Name":"Somu","payment":{"Total":600},"transaction":{"Price":600}});
db.transactions.insertOne({"ID":"2","Name":"Ravi","payment":{"Total":400},"transaction":{"Price":400}});
db.transactions.insertOne({"ID":"3","Name":"Bhuvan","payment":{"Total":320.50},"transaction":{"Price":320.50}});

db.transactions.find({ "Name": "Somu" });

db.transactions.find({ "payment.Total": 600 });

db.transactions.find({ "transaction.Price": { $gte: 300, $lte: 500 } });

db.transactions.aggregate([{$group:{
        _id: null,
        totalAmount: { $sum: "$payment.Total" }
      }
    }]);
```

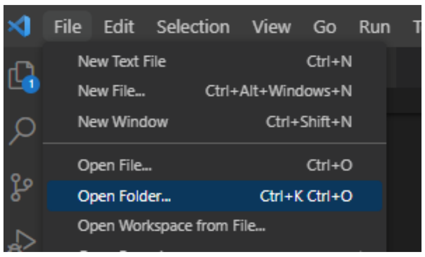
Installing Node.js (ONLY if you want to try this in your Laptop else just skip this part)

Node.js link: https://nodejs.org/en/download/prebuilt-installer/current

VS Code: https://code.visualstudio.com/download

(Install and Restart Computer)

Always while running a new program Create a new folder and open in VS code



Then add Files as necessary

If there is an error while trying out npm. Run VS Code as admistrator and run the command

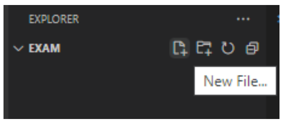
```
Set-ExecutionPolicy RemoteSigned
```

3 a. Write a program to check request header for cookies.

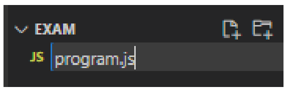
b. write node.js program to print the a car object properties, delete the second property and get length of

the object.

Make a new file



Create folder



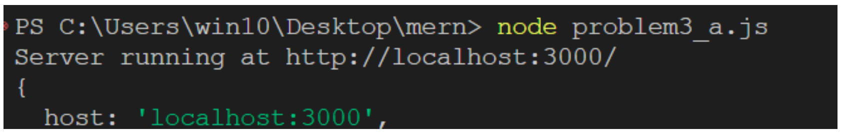
Console installations required(Type in VS terminal):

```
npm i init -y
```

A. Make new file (program_a.js)

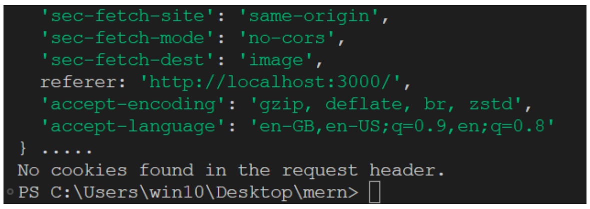
```
const http = require("http");
// Create a server
const server = http.createServer((req, res) => {
  const headers = req.headers;
  console.log(headers, ".....");
  if (headers.cookie) {
    console.log("Cookies:", headers.cookie);
  } else {
    console.log("No cookies found in the request header.");
  }
});

server.listen(3000, () => {
  console.log("Server running at http://localhost:3000/");
});
```

In the terminal type ‘node program_a.js’



```
PS C:\Users\win10\Desktop\mern> node problem3_a.js
Server running at http://localhost:3000/
{
  host: 'localhost:3000',
```

Open ‘http://localhost:3000/’ in a Browser and check the Console again for the output



```
   'sec-fetch-site': 'same-origin',
   'sec-fetch-mode': 'no-cors',
   'sec-fetch-dest': 'image',
   referer: 'http://localhost:3000/',
   'accept-encoding': 'gzip, deflate, br, zstd',
   'accept-language': 'en-GB,en-US;q=0.9,en;q=0.8'
 } .....
 No cookies found in the request header.
 PS C:\Users\win10\Desktop\mern>
```

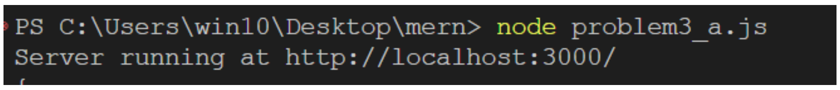
B. Make new file (program_b.js)

```
const express = require("express");
const app = express();

// Sample car object
const car = {
  brand: "Toyota",
  model: "Camry",
  year: 2020,
  color: "Black",
};

// API endpoint to get car object properties
app.get("/api/car", (req, res) => {
  res.json(car);
});

// API endpoint to delete the second property and get object length
app.delete("/api/car/:propertyIndex", (req, res) => {
  const propertyIndex = parseInt(req.params.propertyIndex);

  if (
    isNaN(propertyIndex) ||
    propertyIndex < 0 ||
    propertyIndex >= Object.keys(car).length
  ) {
    return res.status(400).json({ error: "Invalid property index" });
  }

  const propertyToDelete = Object.keys(car)[propertyIndex];
  delete car[propertyToDelete];
  res.json({
    deletedProperty: propertyToDelete,
    remainingProperties: Object.keys(car),
    objectLength: Object.keys(car).length,
  });
});

const port = 3000;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Console installations required:

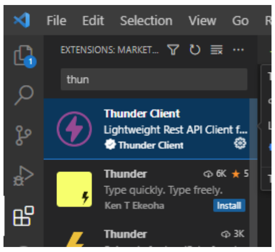
```
npm i init -y

npm i express;
```

In the terminal type ‘node program_b.js’

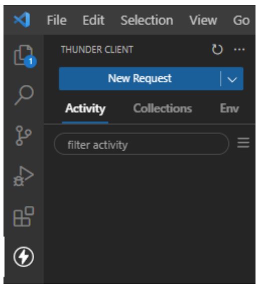


```
PS C:\Users\win10\Desktop\mern> node problem3_a.js
Server running at http://localhost:3000/
```

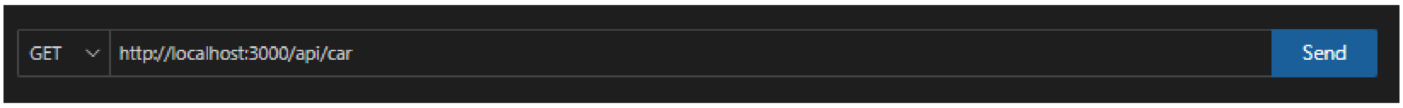
And Install ThunderClient through the Extentions Menu:



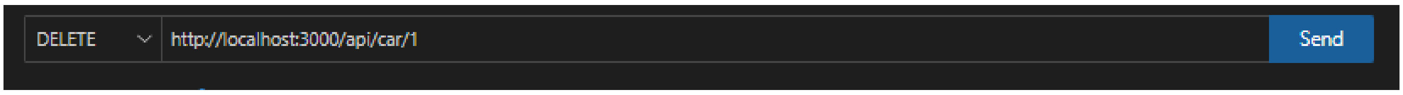
Click on ‘New Request’



For the Outputs:



GET http://localhost:3000/api/car Send



DELETE http://localhost:3000/api/car/1 Send

4 a. Read the data of a student containing usn, name, sem, year_of_admission from node js and store it in

the mongodb

b. For a partial name given in node js, search all the names from mongodb student documents created in Question(a)

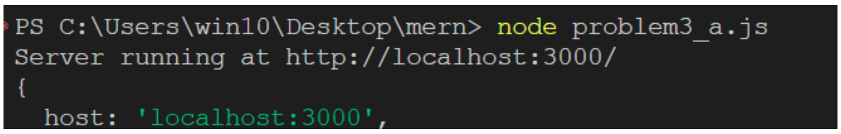
Console Installations required:

```
npm i init -y

npm i mongoose express
```

A. Make new file (program_a.js)

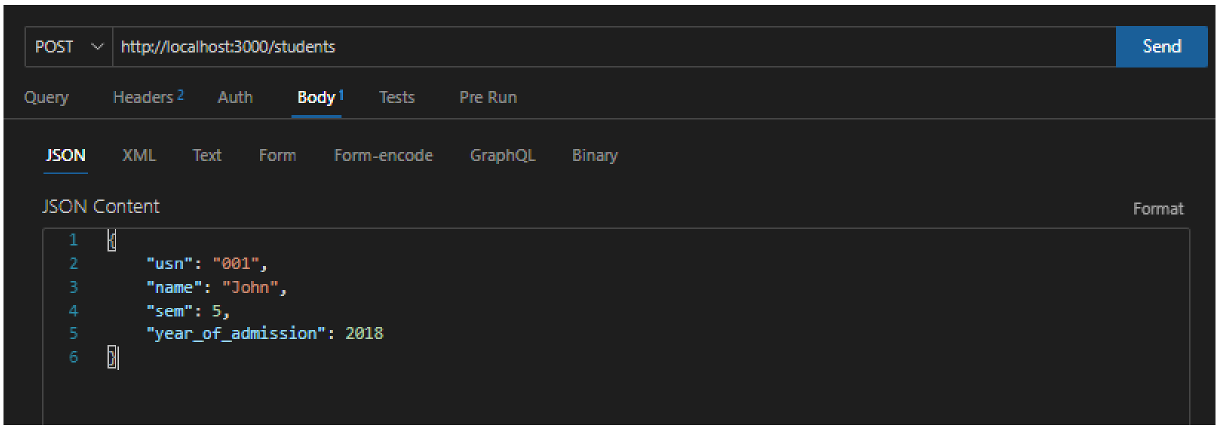
```
const express = require("express");
const mongoose = require("mongoose");

const app = express();
const port = 3000;

mongoose.connect("mongodb://127.0.0.1:27017/COLLEGE", {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => console.log("MongoDB connected"))
.catch(err => console.error("Failed to connect to MongoDB:", err));

// Create a schema for student data
const studentSchema = new mongoose.Schema({
  usn: String,
  name: String,
  sem: Number,
  year_of_admission: Number
});

const Student = mongoose.model("Students", studentSchema);
app.use(express.json());

// API endpoint to store student data in MongoDB
app.post("/students", async (req, res) => {
  try {
    // Read student data from the request body
    const studentData = req.body;

    // Insert student data into MongoDB collection
    const result = await Student.insertMany(studentData);
    console.log(`Inserted ${result.length} documents into the collection`);
    res.status(201).json({ message: `Inserted ${result.length} documents` });
  } catch (err) {
    console.error("Error inserting documents:", err);
    res.status(500).json({ error: "Error inserting documents" });
  }
});

// API endpoint to search for students by partial name
app.get("/students/search", async (req, res) => {
  try {
    const partialName = req.query.partialName;

    if (!partialName) {
      return res.status(400).json({ error: "Partial name parameter is required" });
    }

    const regex = new RegExp(partialName, "i"); // "i" for case-insensitive search
    const students = await Student.find({ name: { $regex: regex } });
    res.status(200).json(students);
  } catch (err) {
    console.error("Error searching for students:", err);
    res.status(500).json({ error: "Error searching for students" });
  }
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

In the terminal type ‘node program.js’



```
PS C:\Users\win10\Desktop\mern> node problem3_a.js
Server running at http://localhost:3000/
{
  host: 'localhost:3000',
```

1.For output use ThunderClient Again: (Make a new request and type both the top address and the json content)

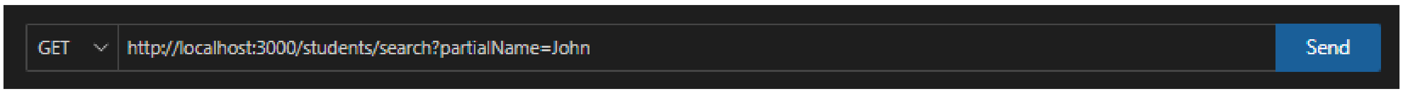


http://localhost:3000/students

JSON Content

```
{
    "usn": "001",
    "name": "John",
    "sem": 5,
    "year_of_admission": 2018
}
```

2.



http://localhost:3000/students/search?partialName=John

5.Implement all CRUD operations on a File System using Node JS

Console Installation Needed:

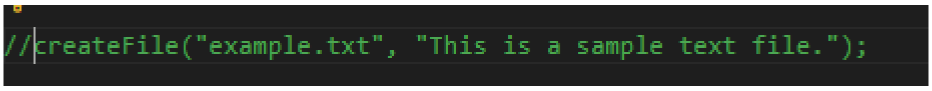
```
npm i init -y

npm i fs
```

Program: Make new file (program.js)

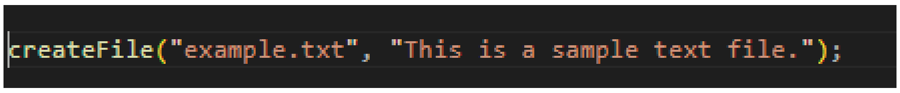
```
// Implement all CRUD operations on a File System using Node JS
//!run one function at a time by removing the ‘//’

const fs = require("fs");

const createFile = (fileName, data) => {
  fs.writeFile(fileName, data, (err) => {
    if (err) {
      console.error("Error creating file:", err);
    } else {
      console.log("File created successfully.");
    }
  });
};

// createFile("example.txt", "This is a sample text file.");

const readFile = (fileName) => {
  fs.readFile(fileName, "utf8", (err, data) => {
    if (err) {
      console.error("Error reading file:", err);
    } else {
      console.log("File content:");
      console.log(data);
    }
  });
};

// readFile("example.txt");

const appendFile = (fileName, newData) => {
  fs.appendFile(fileName, newData, (err) => {
    if (err) {
      console.error("Error appending to file:", err);
    } else {
      console.log("Data appended to file successfully.");
    }
  });
};

// appendFile("example.txt", "\nAdditional data appended.");

const updateFile = (fileName, newData) => {
  fs.writeFile(fileName, newData, (err) => {
    if (err) {
      console.error("Error updating file:", err);
    } else {
      console.log("File updated successfully.");
    }
  });
};

// updateFile("example.txt", "This is updated content.");

const deleteFile = (fileName) => {
  fs.unlink(fileName, (err) => {
    if (err) {
      console.error("Error deleting file:", err);
    } else {
      console.log("File deleted successfully.");
    }
  });
};

// deleteFile("example.txt");
```

For Output:

Remove the // of the comment lines one at a time and run



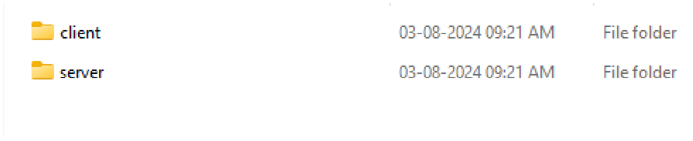
```
//createFile("example.txt", "This is a sample text file.");
```



```
createFile("example.txt", "This is a sample text file.");
```

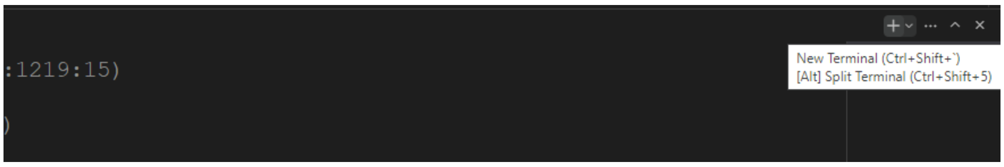
6 Develop the application that sends fruit name and price data from client side to Node.js server using Ajax

In a New folder Create two new folders



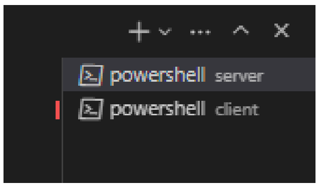
client	03-08-2024 09:21 AM	File folder
server	03-08-2024 09:21 AM	File folder

Create new terminal by clicking the + on the right side of the VS terminal:

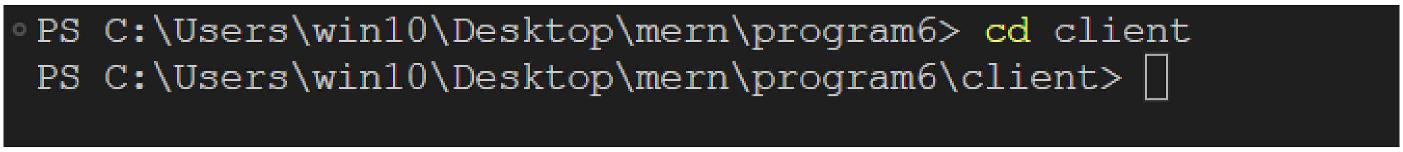


```
:1219:15)
)
```

New Terminal (Ctrl+Shift+`)
[Alt] Split Terminal (Ctrl+Shift+5)

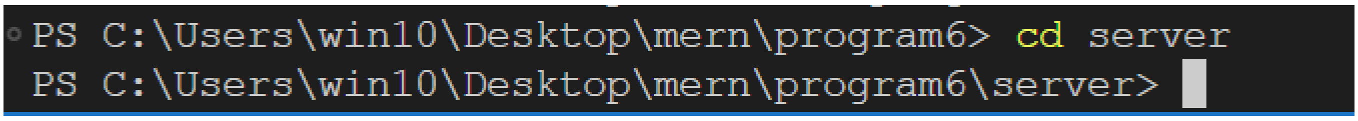


In the 1st VS code Terminal type:



```
PS C:\Users\win10\Desktop\mern\program6> cd client
PS C:\Users\win10\Desktop\mern\program6\client>
```

In the 2nd VS code Terminal type:



```
PS C:\Users\win10\Desktop\mern\program6> cd server
PS C:\Users\win10\Desktop\mern\program6\server>
```

Client Side(App.js):

Console Installation Needed (In the Client terminal type):

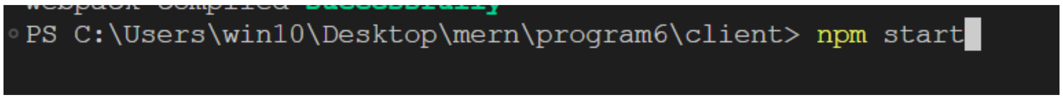
```
npm i init -y

npm i npm -g

npm i npx axios

npx create-react-app Fruit
```

```
cd Fruit
```

Program (Replace code in App.js which is in the client > src folder):

```
import React, { useState } from "react";
import axios from "axios";

function App() {
  const [name, setName] = useState("");
  const [price, setPrice] = useState("");
  const url = "http://localhost:5000";
  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post(`${url}/api/fruits`, { name, price });
      console.log("Data sent successfully");
      // Optionally, you can reset the form fields here
    } catch (error) {
      console.error("Error sending data:", error);
    }
  };

  return (
    <div>
      <h1>Fruit Data Form</h1>
      <form onSubmit={handleSubmit}>
        <label>
          Fruit Name:
          <input
            type="text"
            value={name}
            onChange={(e) => setName(e.target.value)}
          />
        </label>
        <br />
        <label>
          Price:
          <input
            type="text"
            value={price}
            onChange={(e) => setPrice(e.target.value)}
          />
        </label>
        <br />
        <button type="submit">Submit</button>
      </form>
    </div>
  );
}

export default App;
```

Output: (in the Client Terminal run)



```
PS C:\Users\win10\Desktop\mern\program6\client> npm start
```

Server Side(Make new file server.js):

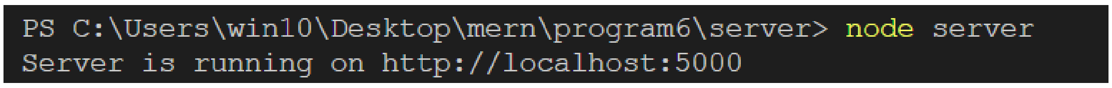
Console Installation Needed:

```
npm i init -y

npm install express body-parser cors axios
```

Program(server.js):

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const app = express();
const PORT = 5000;

app.use(bodyParser.json());
app.use(cors());
// Handle POST request to '/api/fruits'
app.post("/api/fruits", (req, res) => {
  const { name, price } = req.body;
  console.log(`Received data from client: Name - ${name}, Price - ${price}`);
  // Here, you can process the data (e.g., save to database) and send back a response if needed
  res.status(200).send("Data received successfully");
});

app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

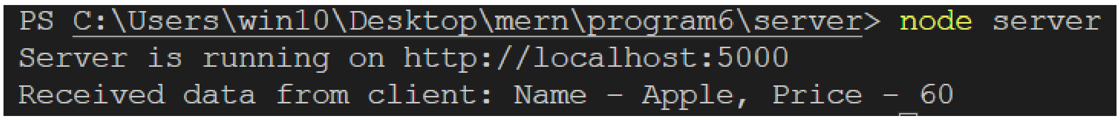
Output(server terminal):



```
PS C:\Users\win10\Desktop\mern\program6\server> node server
Server is running on http://localhost:5000
```

OUTPUT:

Type values in the webpage and submit. It should appear in the Server terminal



```
PS C:\Users\win10\Desktop\mern\program6\server> node server
Server is running on http://localhost:5000
Received data from client: Name - Apple, Price - 60
```

7. Develop an authentication mechanism with email_id and password using HTML and Express JS (POST method)

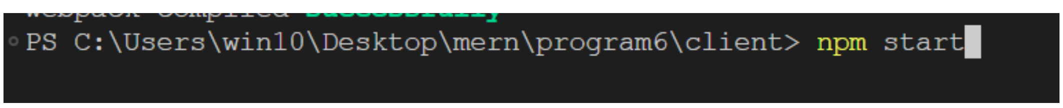
Follow the same Two folder and Two terminal method as Program 6. Create the react app by installing the same npm Commands. And then the App.js and server.js Codes are

Client Side(App.js):

```
import React, { useState } from "react";
import axios from "axios";

function App() {
  const [registerData, setRegisterData] = useState({ email: "", password: "" });
  const [loginData, setLoginData] = useState({ email: "", password: "" });
  const [message, setMessage] = useState("");

  const handleRegister = async (e) => {
    e.preventDefault();
    try {
      await axios.post("http://localhost:3000/register", registerData);
      setMessage("User registered successfully");
    } catch (error) {
      setMessage("Error registering user");
      console.error("Error registering user:", error);
    }
  };

  const handleLogin = async (e) => {
    e.preventDefault();
    try {
      await axios.post("http://localhost:3000/login", loginData);
      setMessage("Login successful");
    } catch (error) {
      setMessage("Invalid credentials");
      console.error("Error logging in user:", error);
    }
  };

  return (
    <div>
      <h2>User Registration</h2>
      <form onSubmit={handleRegister}>
        <label>Email:</label>
        <input type="email"
          value={registerData.email}
          onChange={(e) =>
            setRegisterData({ ...registerData, email: e.target.value })
          }
          required
        />
        <br />
        <label>Password:</label>
        <input type="password"
          value={registerData.password}
          onChange={(e) =>
            setRegisterData({ ...registerData, password: e.target.value })
          }
          required
        />
        <br />
        <button type="submit">Register</button>
      </form>

      <h2>User Login</h2>
      <form onSubmit={handleLogin}>
        <label>Email:</label>
        <input type="email"
          value={loginData.email}
          onChange={(e) =>
            setLoginData({ ...loginData, email: e.target.value })
          }
          required
        />
        <br />
        <label>Password:</label>
        <input type="password"
          value={loginData.password}
          onChange={(e) =>
            setLoginData({ ...loginData, password: e.target.value })
          }
          required
        />
        <br />
        <button type="submit">Login</button>
      </form>

      {message && <p>{message}</p>}
    </div>
  );
```

```
}

export default App;
```

Output(Client Side):



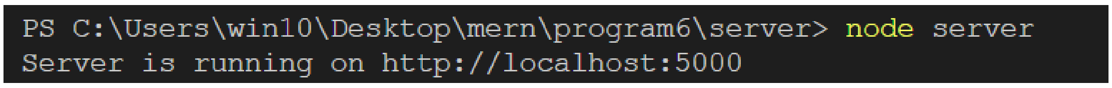
```
PS C:\Users\win10\Desktop\mern\program6\client> npm start
```

Server Side(server.js):

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");

const app = express();
const PORT = 3000;

app.use(bodyParser.json());
app.use(cors());
// Sample user data (replace this with your database)
const users = [];

// Route to handle user registration
app.post("/register", async (req, res) => {
  try {
    const { email, password } = req.body;

    // Check if email is already registered
    const existingUser = users.find((user) => user.email === email);
    if (existingUser) {
      return res.status(400).send("Email already exists");
    }

    // Store the user data (in memory for now, replace this with database storage)
    users.push({ email, password });

    res.status(201).send("User registered successfully");
  } catch (error) {
    console.error("Error registering user:", error);
    res.status(500).send("Internal server error");
  }
});

// Route to handle user login
```

```
app.post("/login", async (req, res) => {
  try {
    const { email, password } = req.body;

    // Find the user by email
    const user = users.find((user) => user.email === email);
    if (!user) {
      return res.status(401).send("Invalid credentials");
    }

    // Compare the password
    const passwordMatch = users.find((user) => user.password === password);
    if (!passwordMatch) {
      return res.status(401).send("Invalid credentials");
    }

    res.status(200).send("Login successful");
  } catch (error) {
    console.error("Error logging in user:", error);
    res.status(500).send("Internal server error");
  }
});

// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Output(server terminal):

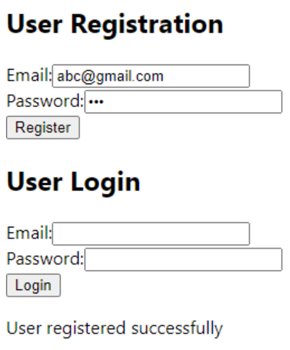


```
PS C:\Users\win10\Desktop\mern\program6\server> node server
Server is running on http://localhost:5000
```

OUTPUT:

Refresh the page (stop and start the server if any problems)

And type register email and password and hit register



User Registration

Email:	abc@gmail.com
Password:	•••

Register

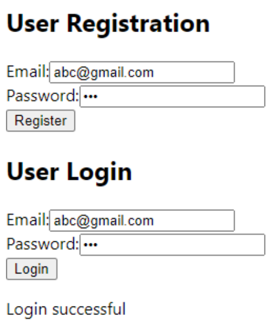
User Login

Email:	
Password:	

Login

User registered successfully

Do the same email and password for login



User Registration

Email: abc@gmail.com
Password: •••
Register

User Login

Email: abc@gmail.com
Password: •••
Login

Login successful

8 a. Develop two routes: find_prime_100 and find_cube_100 which prints prime numbers less than 100 and cubes less than 100 using Express JS routing mechanism

Console installations required:

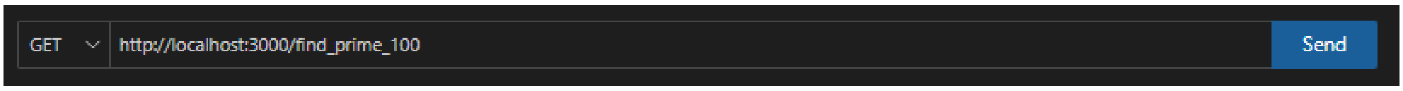
```
npm i init -y

npm i express;
```

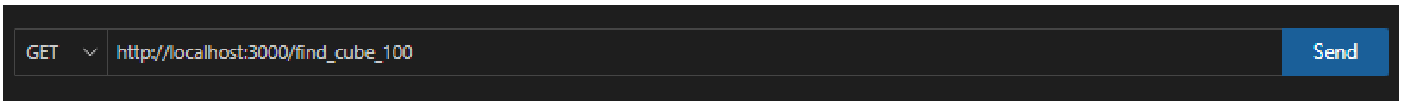
Program(program.js):

```
const express = require("express");
const app = express();
const PORT = 3000;

// Route to find prime numbers less than 100
app.get("/find_prime_100", (req, res) => {
  const primes = findPrimes(100);
  res.json({ primes });
});

// Route to find cubes less than 100
app.get("/find_cube_100", (req, res) => {
  const cubes = findCubes(100);
  res.json({ cubes });
});

// Function to find prime numbers less than n
function findPrimes(n) {
  const primes = [];
  for (let i = 2; i < n; i++) {
    let isPrime = true;
    for (let j = 2; j <= Math.sqrt(i); j++) {
      if (i % j === 0) {
        isPrime = false;
        break;
      }
    }
    if (isPrime) {
      primes.push(i);
    }
  }
  return primes;
}

// Function to find cubes less than n
function findCubes(n) {
  const cubes = [];
  for (let i = 1; i < n; i++) {
    const cube = i * i * i;
    if (cube < n) {
      cubes.push(cube);
    } else {
      break;
    }
  }
  return cubes;
}

// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Output:

In the ThunderClient(make new request) or Browser Address Bar type


GET http://localhost:3000/find_prime_100


GET http://localhost:3000/find_cube_100

9 a. Develop a React code to build a simple search filter functionality to display a filtered list based on the search query entered by the user.

Console Installation Needed (In the terminal type):

```
npm i init -y
npm i npm -g

npm i npx axios

npx create-react-app search

cd search
```

Program(App.js):

```
import React, { useState } from "react";

function App() {
  // Define the JSON data directly in the program
  const jsonData = ["Apple", "Banana", "Orange", "Pineapple", "Mango", "Grapes"];
  const [searchQuery, setSearchQuery] = useState("");
  const [filteredItems, setFilteredItems] = useState([]);

  // Function to handle search input change
  const handleSearchChange = (e) => {
    const query = e.target.value.toLowerCase();
    setSearchQuery(query);
    const filtered = jsonData.filter((item) =>
      item.toLowerCase().includes(query)
    );
    setFilteredItems(filtered);
  };

  return (
    <div>
      <h2>Search Filter</h2>
      <input
        type="text"
        placeholder="Search..."
        value={searchQuery}
        onChange={handleSearchChange}
      />
      <ul>
        {filteredItems.map((item, index) => (
          <li key={index}>{item}</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

```
```

Output:

Just type ‘a’ in the search bar in the webpage

10 a. Develop a React code to collect data from rest API.

Console Installation Needed (In the terminal type):

```
npm i init -y

npm i npm -g

npm i npx axios

npx create-react-app data

cd data
```

Program(App.js):

```
import React, { useState, useEffect } from "react";
import axios from "axios";

function DataCollector() {
  const [data, setData] = useState([]);

  useEffect(() => {
    fetchData();
  }, []);

  const fetchData = async () => {
    try {
      const response = await axios.get(
        "https://dummy.restapiexample.com/api/v1/employees"
      );
      setData(response.data.data);
    } catch (error) {
      console.error("Error fetching data:", error);
    }
  };

  return (
    <div>
      <h2>Data Collection</h2>
      <button onClick={fetchData}>Fetch Data</button>
      <ul>
        {data.map((item, index) => (
          <li key={index}>{item.employee_name}</li>
        ))}
      </ul>
    </div>
  );
}

export default DataCollector;
```