1) Write a program to create a simple calculator Application using React JS.

Commands to run in the terminal:

- **npm create-react-app calculator**
- **cd calculator**
- **npm i**
- **npm start**

APP.js

```
import React, { useState } from 'react';
import './App.css';

function App() {
  const [input, setInput] = useState('');
  const [result, setResult] = useState('');

  const handleButtonClick = (value) => {
    if (value === '=') {
      try {
        setResult(eval(input) || 'Error');
      } catch (error) {
        setResult('Error');
      }
    } else if (value === 'C') {
      setInput('');
      setResult('');
    } else {
      setInput((prevInput) => prevInput + value);
    }
  };

  return (
    <div className="calculator">
      <input
        type="text"
        className="calculator-screen"
        value={input}
        disabled
      />
      <div className="calculator-buttons">
        <button onClick={() => handleButtonClick('7')}>7</button>
        <button onClick={() => handleButtonClick('8')}>8</button>
        <button onClick={() => handleButtonClick('9')}>9</button>
        <button onClick={() => handleButtonClick('+')}>+</button>
        <button onClick={() => handleButtonClick('4')}>4</button>
        <button onClick={() => handleButtonClick('5')}>5</button>
        <button onClick={() => handleButtonClick('6')}>6</button>
```

```
        <button onClick={() => handleButtonClick('-')}>-</button>
        <button onClick={() => handleButtonClick('1')}>1</button>
        <button onClick={() => handleButtonClick('2')}>2</button>
        <button onClick={() => handleButtonClick('3')}>3</button>
        <button onClick={() => handleButtonClick('')}></button>
        <button onClick={() => handleButtonClick('0')}>0</button>
        <button onClick={() => handleButtonClick('.')}>.</button>
        <button onClick={() => handleButtonClick('=')}>=</button>
        <button onClick={() => handleButtonClick('/')}>/</button>
        <button onClick={() => handleButtonClick('C')}>C</button>
      </div>
      <div className="calculator-result">{result}</div>
    </div>
  );
}

export default App;
```

App.css

```
.calculator {
  width: 300px;
  margin: auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
  background-color: #f9f9f9;
}

.calculator-screen {
  width: 100%;
  margin-bottom: 10px;
  padding: 10px;
  font-size: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.calculator-buttons {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: 10px;
}

.calculator-buttons button {
  padding: 10px;
  font-size: 18px;
  border: none;
```

```
  border-radius: 5px;
  background-color: #fff;
  cursor: pointer;
}

.calculator-buttons button:hover {
  background-color: #f0f0f0;
}

.calculator-result {
  margin-top: 10px;
  font-size: 24px;
  font-weight: bold;
}
```

2) You are a waiter of Meghna's restaurant, your task is to take order from customer (Client) and give it to the chef (Server), create a form using reactjs to take order and create an API which sends those details to the server, use express js and http method. Also Take care of CORS (Cross-Origin Resource Sharing) issue. Note: - Add "Meghnas Resturant" on the top of the webpage, then create a form after leaving margin of atleast 10 px, add basic styling

Solution:

Terminal commands:

Server side:

1. Create index.js
2. npm I express
3. npm init (add the content in package.json)
4. npm i
5. npm run start

client side:

1. npx create-react-app .
2. replace the app.css and app.js file
3. npm i
4. npm start

Index.js: (server side)

```
const express=require('express')
const cors=require('cors')
```

```
const app=express();
app.use(express.json())
//handle cors issue
app.use(cors())

//create order api

app.post('/order',(req,res)=>{
    console.log(req.body)
})

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

Package.json (server side):

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "nodeman": "^1.1.2",
    "nodemon": "^3.0.3"
  }
}
```

App.css (client side): (optional)

```
.heading{
    display: flex;
    justify-content: center;
    align-items: center;
    margin-bottom: 20px;
}
.container{
```

```
    display: flex;
    align-items: center;
    justify-content: center;
}
.form{
    border: 2px solid red;
    padding: 10px;
    height: 500px;
    width: 500px;
    border-radius: 10px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
.form .inputs{
    margin: 5px;
    padding: 10px;
    display: flex;
    flex-direction: column;

}
.form input{
    padding: 10px;
    margin: 5px;
}
```

app. Jsx (client side):

```
import { useState } from 'react'
import './App.css'

function App() {
      const [itemName, setItemName] = useState("")
      const [quantity,setQuantity]=useState("");
      const [instructions, setInstruction]=useState("");

      const order={
        'itemName':itemName,
        'quantity':quantity,
        'instruction':instructions
      }
      const handleSubmit =async (e)=>{
                e.preventDefault();
                 await fetch('http://localhost:3000/order', {
                  method:'POST',
                  headers:{
                    'Content-Type': 'application/json'
```

```
                },
                body:JSON.stringify(order)
              })
    }

  return (
   <>
      <div className='heading'>
         <h1> Meghnas Biryani</h1>
      </div>
      <div className="container">
      <form className="form" onSubmit={handleSubmit}>
           <div className='inputs'>
           <label htmlFor="itemName">Item Name:</label>
           <input type="text" id='itemName' placeholder='Enter Item
Name'  value={itemName} onChange={(e)=>{
              setItemName(e.target.value);
           }} />
           </div>
           <div className='inputs'>
           <label htmlFor="quantity">Quantity:</label>
           <input type="number" id='quantity' placeholder='0' value={quantity}
onChange={(e)=>{
             setQuantity( e.target.value);
           }}/>
           </div>
           <div className='inputs'>
           <label htmlFor="special">Special Instructions:</label>
           <input type="text" id='special' placeholder='Add instructions'
value={instructions} onChange={(e)=>{
             setInstruction(e.target.value)
           }}/>
           </div>

           <button type='submit'>Place Order</button>
      </form>
      </div>
   </>
  )
}

export default App
```

3) Create a json object and add 5 students on that (student details, like Name, USN, Hobbie), using http method and express js. Display those details on the frontend (created using reactjs) and connect with Mongodb database.

Create folders:

- Client
- Server
    - Add file:
        - Index.js
        - Student.json

Terminal commands:

Server side:

1. Create index.js
2. npm I express
3. npm I cors
4. npm I mongoose
5. npm init (add the content in package.json)
6. npm i
7. npm run start

client side:

5. npm create vite@latest.
6. replace the app.css and app.jsx file
7. npm i
8. npm run dev

Index.js (server side):

```
const express= require('express')
const cors=require('cors')
const students=require('./student.json')
const app=express()

app.use(cors())
app.use(express.json())

app.get('/studentdetails',(req,res)=>{
```

```
        if(students){
            res.json(students)
        }else{
            res.json({"error":"No students found"})
        }

})

app.listen(3000,()=>{
    console.log("Server is listening on port 3000");
})
```

Package.json (server side):

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
   "start":"nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "nodemon": "^3.0.3"
  }
}
```

Student.json (server side):

```
[
    {
        "name":"Faisal Tanveer Khan",
        "usn":"1ds21ai099",
        "hobbie":"Time waste"
    },
    {
        "name":"Anamika Jha",
        "usn":"1ds21ai097",
        "hobbie":"Reading books"
    },
    {
        "name":"Shadan Siddique",
```

```
        "usn":"1ds21ai095",
        "hobbie":"Youtube video creation"
    },
    {
        "name":"Rohan Kumar",
        "usn":"1ds21ai093",
        "hobbie":"Stand-up comedy"
    },
    {
        "name":"Ujjwala Sinha",
        "usn":"1ds21ai091",
        "hobbie":"Painting"
    }
]
```

App.css (client side):

```css
.container {
  max-width: 800px;
  margin: 0 auto;
  padding: 20px;
}

.card {
  border: 1px solid #ccc;
  border-radius: 8px;
  padding: 20px;
  margin-bottom: 20px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.card-title {
  font-size: 20px;
  margin-bottom: 10px;
}

.card-text {
  font-size: 16px;
  margin-bottom: 5px;
}
```

App. Jsx (client side):

```jsx
import { useState } from 'react'
import './App.css'

function App() {
  const [students,setStudents]=useState([])

  const studentdetail=async()=>{
```

```
    const res=await fetch('http://localhost:3000/studentdetails',{
       method:'GET',
       headers:{
         'content-type':'application/json'
       }
     })
     const data=await res.json();
     setStudents(data)

  }

 studentdetail()

  // console.log(students)

  return (
    <div className="container">
    <h1 className="mt-4 mb-4">Student Dashboard</h1>
    <div className="row">
      {students.map((student, index) => (
        <div key={index} className="col-lg-4 mb-4">
          <div className="card">
            <div className="card-body">
              <h5 className="card-title">{student.name}</h5>
              <p className="card-text">USN: {student.usn}</p>
              <p className="card-text">Hobbie: {student.hobbie}</p>
            </div>
          </div>
        </div>
      ))}
    </div>
  </div>
  )
}

export default App
```

4) Write a program using JavaScript to test whether the given numbers are even or odd. and display greatest number among three entered numbers by user.

Program:

Index.js:

```
function checkEvenOrOdd(number) {
    return number % 2 === 0 ? 'Even' : 'Odd';
}

function findGreatestNumber(num1, num2, num3) {
    return Math.max(num1, num2, num3);
}

const num1 = parseInt(process.argv[2]);
const num2 = parseInt(process.argv[3]);
const num3 = parseInt(process.argv[4]);

console.log(`Number ${num1} is ${checkEvenOrOdd(num1)}.`);
console.log(`Number ${num2} is ${checkEvenOrOdd(num2)}.`);
console.log(`Number ${num3} is ${checkEvenOrOdd(num3)}.`);

const greatestNumber = findGreatestNumber(num1, num2, num3);
console.log(`The greatest number among ${num1}, ${num2}, and ${num3} is
${greatestNumber}.`);
```

commands in the terminal:

node file_name.js num1 , num2 . num3

example:

- **node index.js 12 13 14**

5) Write a program to input your name and age using form and know whether he/she is eligible to vote or not. [Use function and onclick event] and CSS styles.

Terminal commands:

client side:

1. npm create vite@latest.
2. Replace package.json
3. replace the app.css and app.jsx file
4. npm i
5. npm run dev

Package.json:

```
{
  "name": "vote",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-
warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "devDependencies": {
    "@types/react": "^18.2.55",
    "@types/react-dom": "^18.2.19",
    "@vitejs/plugin-react": "^4.2.1",
    "eslint": "^8.56.0",
    "eslint-plugin-react": "^7.33.2",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.5",
    "vite": "^5.1.0"
  }
}
```

App.css: (optional)

```
.container {
  max-width: 400px;
  margin: 50px auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.form-group {
  margin-bottom: 20px;
}

label {
```

```
  font-weight: bold;
}

.input {
  width: 100%;
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.btn-primary {
  display: block;
  width: 100%;
  padding: 10px;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.btn-primary:hover {
  background-color: #0056b3;
}

.message {
  margin-top: 20px;
  font-size: 18px;
  font-weight: bold;
}
```

App.jsx:

```
import  { useState } from 'react';
import './App.css';

function App() {
  const [name, setName] = useState('');
  const [age, setAge] = useState('');
  const [message, setMessage] = useState('');

  const checkEligibility = () => {
    if (age >= 18) {
      setMessage(`you are eligible to vote!`);
    } else {
      setMessage(`you are not eligible to vote yet.`);
    }
```

```
  };

  return (
    <div className="container">
      <h1>Check Your Voting Eligibility</h1>
      <form>
        <div className="form-group">
          <label htmlFor="name">Name:</label>
          <input
            type="text"
            className="form-control"
            id="name"
            value={name}
            onChange={(e) => setName(e.target.value)}
          />
        </div>
        <div className="form-group">
          <label htmlFor="age">Age:</label>
          <input
            type="number"
            className="form-control"
            id="age"
            value={age}
            onChange={(e) => setAge(e.target.value)}
          />
        </div>
        <button
          type="button"
          className="btn btn-primary"
          onClick={checkEligibility}
        >
          Check Eligibility
        </button>
      </form>
      {message && <p className="message">{message}</p>}
    </div>
  );
}

export default App;
```

6) Create database, create collection, insert data, find, find one, sort, limit, skip, distinct, projection Questions: i. 1.display all the documents ii. 2.Display all the students in BCA iii. 3.Display all the students in ascending order iv. 4.Display first 5 students v. 5.display students 5,6,7 vi. 6.list the degree of student "Rahul" vii. 7.Display students details of 5,6,7 in descending order of percentage viii. 8.Display the number of students in BCA

Solution:

Go to cmd > mongosh > it opens “test >”

To Type the highlighted in the question.

1.use aimldb -> (your database name)

2. show collections or create one as student or student1

3. use student1

4. then type highlighted one
db.stud1col1.insert({srn:110,sname:"Rahul",degree:"BCA",sem:6,CGPA:7.9})

Give 10 more inputs

db.student1.insertMany([{ srn: 110, sname: "Rahul", degree: "BCA", sem: 6, CGPA: 7.9 }, { srn: 111, sname: "Sara", degree: "BCA", sem: 5, CGPA: 8.5 }, { srn: 112, sname: "John", degree: "BBA", sem: 4, CGPA: 8.0 }, { srn: 113, sname: "Emma", degree: "B.Tech", sem: 7, CGPA: 7.2 }, { srn: 114, sname: "Mike", degree: "BCA", sem: 8, CGPA: 7.8 }, { srn: 115, sname: "Alice", degree: "B.Tech", sem: 6, CGPA: 8.9 }, { srn: 116, sname: "Tom", degree: "BCA", sem: 5, CGPA: 7.4 }, { srn: 117, sname: "Sophia", degree: "BBA", sem: 6, CGPA: 8.2 }, { srn: 118, sname: "Chris", degree: "B.Tech", sem: 4, CGPA: 6.9 }, { srn: 119, sname: "Eva", degree: "BCA", sem: 7, CGPA: 8.7 }])

Then queries:

1. db.student1.find()

2. db.student1.find({ degree: "BCA" })

3. db.student1.find().sort({ CGPA: 1 })

4. db.student1.find().limit(5)

5. db.student1.find().skip(4).limit(3)

6. db.student1.findOne({ sname: "Rahul" }, { degree: 1, _id: 0 })

7. db.student1.find().skip(4).limit(3).sort({ CGPA: -1 })

8. db.student1.find({ degree: "BCA" }).count()

7) Write an API to fetch a local json file using fetch () function and print the contents of the json file on front end also connect with Mongodb database.Note:Execute using REST)

Terminal commands:

Server side:

1) Create index.js
2) npm I express
3) npm init (add the content in package.json)
4) npm i
5) npm run start

client side: (app.jsx)

1) npm create vite@latest.
2) replace the app.css and app.jsx file
3) npm i
4) npm run dev

client side: (app.js)

1) npm create-react-app client
2) replace the app.css and app.js file
3) npm i
4) npm start

Create folders:

- Client
- Server
    - Add file:
        - Index.js
        - data.json

Server-side:

Package.json (server side):

```
{
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.3",
    "mongoose": "^8.2.0"
  },
  "scripts": {
    "start": "node server.js"
  },
  "name": "server",
  "version": "1.0.0",
  "main": "server.js",
  "author": "",
  "license": "ISC",
  "keywords": [],
  "description": ""
}
```

data.json (server-side):

```
[{
    "name":"Suprith",
    "usn":"52",
    "hobby":"Playing"
},
{
    "name":"sup",
    "usn":"12",
    "hobby":"killing"
}

]
```

Index.js (server-side):

```
const express= require('express')
const cors=require('cors')
const data=require('./data.json')
const app=express()

app.use(cors())
app.use(express.json())

const mongoose=require('mongoose')
mongoose.connect('mongodb://localhost:27017/mydatabase').then(()=>{
    console.log("connected to database");
})

app.get('/data',(req,res)=>{
         res.json(data)
})

app.listen(3000,()=>{
    console.log("Server is listening on port 3000");
})
```

APP.js or App.jsx (server-side):

```
import React, { useState, useEffect } from 'react';
import './App.css'; // Import CSS file

function App() {
  const [jsonData, setJsonData] = useState(null);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await fetch('http://localhost:3000/data');
        const data = await response.json();
        setJsonData(data);
      } catch (error) {
        console.error('Error fetching data:', error);
      }
    };

    fetchData();
  }, []);

  return (
    <div className="container"> {/* Apply container class */}
      <div className="data-container"> {/* Apply data-container class */}
```

```
        <h2>JSON Data</h2>
        <pre>{JSON.stringify(jsonData, null, 2)}</pre>
      </div>
    </div>
  );
}

export default App;
```

APP.css (client-side):

```
.container {
  max-width: 800px;
  margin: 0 auto;
  padding: 20px;
}

.data-container {
  background-color: #f0f0f0;
  border-radius: 5px;
  padding: 20px;
  margin-top: 20px;
}

h2 {
  margin-bottom: 10px;
}

pre {
  background-color: #e0e0e0;
  padding: 10px;
  border-radius: 5px;
  overflow-x: auto;
  font-family: 'Courier New', Courier, monospace;
}
```

8) Write a lab program that guides students through setting up a MongoDB database and establishing a connection to it from a React project. The program should cover the following steps: Install MongoDB locally or use a cloud-based MongoDB service. Create a new database and collection in MongoDB. Set up a React project using Create React App or a similar tool. Install necessary packages (e.g., mongoose) to enable MongoDB connectivity in the React project. Write code to establish a connection to the MongoDB database from the React project. Implement a simple query to retrieve data from MongoDB and display it in the React project

Terminal commands:

Server side:

1. Create index.js
2. npm I express
3. npm init (add the content in package.json)
4. npm i
5. npm run start

client side: (app.js)

1. npm create-react-app client
2. replace the app.css and app.js file
3. npm i
4. npm start

Create folders:

- Client
- Server
    - Add file:
        - Index.js
        - data.json

Index.js (server-side):

```
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');

const app = express();
app.use(express.json());
app.use(cors());

// Database connection
mongoose.connect('mongodb://localhost:27017/my_database', {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(() => {
    console.log("Connected to database");
}).catch(err => {
    console.error("Connection to database failed:", err);
});

// Schema
const userSchema = new mongoose.Schema({
    id: Number,
    name: String
});

const User = mongoose.model('User', userSchema);

// Route to get users
app.get('/users', async (req, res) => {
    try {
        const users = await User.find();
        res.json(users);
    } catch (error) {
        console.error("Error fetching users:", error);
        res.status(500).json({ error: "Internal Server Error" });
    }
});

// Start the server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Server started on port ${PORT}`);
});
```

Package.json: (server-side)

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.3",
    "mongoose": "^8.2.0"
  }
}
```

APP.CSS (client-side)

```
.container {
  max-width: 600px;
  margin: 10px;
  padding: 20px;
}

h1 {
  text-align: center;
  font-size: 24px;
  margin-bottom: 20px;
}

ul {
  list-style-type: none;
  padding: 0;
}

li {
  background-color:#f9f9f9;
  padding: 10px;
  margin-bottom: 5px;
  border-radius: 5px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

li:hover {
  background-color: #e0e0e0;
```

```
}
```

APP.js (client-side):

```
import React, { useEffect, useState } from 'react';

function App() {
    const [users, setUsers] = useState([]);

    useEffect(() => {
        const fetchData = async () => {
            try {
                const response = await fetch('http://localhost:3000/users');
                const users = await response.json();
                setUsers(users);
            } catch (error) {
                console.error("Error fetching users:", error);
            }
        };

        fetchData();
    }, []);

    return (
        <div>
            <h1>Users</h1>
            <ul>
                {users.map(user => (
                    <li key={user.id}>{user.name}</li>
                ))}
            </ul>
        </div>
    );
}

export default App;
```