2. Weather App

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Weather App</title>
</head>
<body>
    <label for="cityInput">Enter City Name:</label>
    <input type="text" id="cityInput">
    <button onclick="getWeather()">Get Weather</button>
    <div id="weatherDetails"></div>

    <script>
        function getWeather() {
            const city = document.getElementById("cityInput").value;
            const xhr = new XMLHttpRequest();

            xhr.onload = function() {
                if (xhr.status === 200) {
                    const data = JSON.parse(xhr.responseText);
                    document.getElementById("weatherDetails").innerHTML =
`City: ${data.name}<br>Temperature: ${data.main.temp}°C<br>Condition:
${data.weather[0].description}`;
                }
            };

            xhr.open("GET",
`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=93f26e3c57081a
6210de53b8dcfdfea4&units=metric`, true);
            xhr.send();
        }
    </script>
</body>
</html>
```

3. Node server at port

```
var http = require('http');
var server = http.createServer(function (req, res) {
    if (req.url == '/') {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write('This is home Page. ');
        res.end();
    }
    else res.end('Invalid Request!');
});
server.listen(8000);
console.log('Node.js web server at port 8000 is running..')
```

4. Read from a file and display

```
var fs = require('fs');
try {
    var data = fs.readFileSync('my-file.txt', 'utf8');
    console.log(data);
} catch (e) {
    console.log('Error:', e.stack);
}
```

5. File exists, append, or create and write

```
const fs = require('fs');
const readline = require('readline').createInterface({
    input: process.stdin, output: process.stdout
});

readline.question('Enter file name: ', (fileName) => {
    readline.question('Enter text to append: ', (text) => {
        fs.appendFile(fileName, text, (err) => {
            if (err) {
                fs.writeFile(fileName, text, (err) => {
                    if (err) {
                        console.error(err);
                        return;
                    }
                    console.log('File created');
                });
                return;
            } console.log('Text appended to the file.');
        });
        readline.close();
    });
});
```

7. Create a form such that, based on student roll number provided by user, the student details should be fetched (using ExpressJS)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Student Details Form</title>
</head>
<body>
    <h2>Fetch Student Details</h2>
    <div>
        <form action="/students" method="GET">
            <div>
                <label for="roll">Enter Roll Number:</label>
                <input type="text" id="roll" name="roll" required>
            </div>
            <button type="submit">Fetch Details</button>
        </form>
        <table>
            <tbody id="data"></tbody>
        </table>
    </div>
    <script>
        document.forms[0].addEventListener('submit', async function (e) {
            e.preventDefault();
            const res = await
fetch(`http://localhost:3000/students/${roll.value}`);
            const s = await res.json();
            const data = document.getElementById('data');
            data.innerHTML = '';
            if (s) {
                data.innerHTML = `
                    <tr><td>Name</td><td>${s.name}</td></tr>
                    <tr><td>Roll Number</td><td>${s.roll}</td></tr>
                    <tr><td>Age</td><td>${s.age}</td></tr>
                    <tr><td>Grade</td><td>${s.grade}</td></tr>
                `;
            } else {
                data.innerHTML = '<tr><td colspan="2">Student not
found</td></tr>';
            }
        });
    </script>
</body>

</html>
```

JS:

```
const express = require('express');
const app = express();
const port = 3000;

app.use(require('cors')())

const students = [
    { name: "John Doe", roll: "123", age: 20, grade: "A" },
    { name: "Jane Smith", roll: "456", age: 22, grade: "B" },
];

app.get('/students/:roll', (req, res) => {
    const roll = req.params.roll;
    const student = students.find(s => s.roll === roll);

    if (student) {
        res.json(student);
    } else {
        res.status(404).json({ error: 'Student not found' });
    }
});

app.get('/', (req, res) => {
    return res.sendFile(__dirname + '/ex5.html');
});

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

6. Student database in MongoDB with all the details of students

```
show dbs;

use student;

insert into studentinfo collection

db.studentinfo.insert({name:"john", id:"20bd1a05051",
course:"b.tech", branch:"cse"})

db.studentinfo.insert({name:"reena", id:"20bd1a0502",
course:"M.tech", branch:"it"})

db.studentinfo.find({})
```

8. CRUD Student DB

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

dotenv.config();

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

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

async function connect() {
    console.log('Connected to MongoDB initiated');
    await mongoose.connect(process.env.MONGODB_URI)
    console.log('Connected to MongoDB');
}

connect()

const studentSchema = new mongoose.Schema({
    name: String,
    rollNumber: { type: String, unique: true },
    age: Number,
    grade: String,
});

const Student = mongoose.model('Student', studentSchema);

app.get('/students/:rollNumber', async (req, res) => {
    try {
        const rollNumber = req.params.rollNumber;
        const student = await Student.findOne({ rollNumber });
        res.json(student);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

app.post('/students', async (req, res) => {
    try {
        const student = new Student(req.body);
        await student.save();
        res.json(student);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});
```

```
app.get('/students', async (req, res) => {
    try {
        const students = await Student.find();
        res.json(students);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

app.put('/students/:rollNumber', async (req, res) => {
    try {
        const rollNumber = req.params.rollNumber;
        await Student.findOneAndUpdate({ rollNumber }, req.body);
        res.json({ message: 'Student updated successfully' });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

app.delete('/students/:rollNumber', async (req, res) => {
    try {
        const rollNumber = req.params.rollNumber;
        await Student.findOneAndDelete({ rollNumber });
        res.json({ message: 'Student deleted successfully' });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

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

1. J. Student data form table

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Student Information</title>
</head>

<body>
    <h2>Student Information</h2>
    <form id="studentForm">
        <label for="name">Name:</label>
        <input type="text" id="uname" name="uname"><br>
        <label for="rollNo">Roll Number:</label>
        <input type="text" id="rollNo" name="rollNo"><br>
        <label for="marks">Marks:</label>
        <input type="number" id="marks" name="marks"><br>
        <button type="button" onclick="submitForm()">Submit</button>
    </form>
    <table id="resultTable" border="2">
        <tr>
            <td>Name</td>
            <td>Roll</td>
            <td>Marks</td>
            <td>GPA</td>
        </tr>
    </table>
    <script>
        function submitForm() {
            const gpa = marks.value / 10;
            const tableRow = document.createElement('tr');
            tableRow.innerHTML = `
                    <td>${uname.value}</td>
                    <td>${rollNo.value}</td>
                    <td>${marks.value}</td>
                    <td>${gpa.toFixed(2)}</td>
                `;
            document.getElementById('resultTable').appendChild(tableRow);
        }
    </script>
</body>

</html>
```

1. i. JSON Display HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Data</title>
</head>
<body>
    <h2>Student Data</h2>
    <script>
        fetch('s1.json')
            .then(response => response.json())
            .then(jsonData => {
                const tableHTML = `
                    <table border='2'>
                        <thead>
                            <tr>
                                ${Object.keys(jsonData.student[0]).map(key => `<th>${key}</th>`).join('')}
                            </tr>
                        </thead>
                        <tbody>
                            ${jsonData.student.map(student => `
                                <tr>
                                    ${Object.values(student).map(value => `<td>${value}</td>`).join('')}
                                </tr>
                            `).join('')}
                        </tbody>
                    </table>
                `;
                document.body.innerHTML += tableHTML;
            })
            .catch(error => console.error('Error fetching JSON:', error));
    </script>
</body>

</html>
```

1. a. SwapCase

```
const readline = require('readline');
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});
function swapCase(instr) {
    var ss = '';
    for (var i = 0; i < instr.length; i++) {
        var ch = instr[i];
        if (ch === ch.toUpperCase()) {
            ss += ch.toLowerCase();
        } else {
            ss += ch.toUpperCase();
        }
    }
    return ss;
}
rl.question('Enter a string: ', function (input) {
    var output = swapCase(input);
    console.log('Swapped case:', output);
    rl.close();
});
```

1. b. Frequency

```
var arr1 = [3, 'a', 'a', 'a', 2, 3, 'a', 3, 'a', 2, 4, 9, 3];
var mf = 1;
var m = 0;
var item;
for (var i = 0; i < arr1.length - 1; i++) {
    for (var j = i; j < arr1.length; j++) {
        if (arr1[i] == arr1[j])
            m++;
        if (mf < m) {
            mf = m;
            item = arr1[i];
        }
    }
    m = 0;
}
console.log(item + " ( " + mf + " times ) ");
```

1. c. Remove Duplicates

```
function remDup(arr) {
    var arr1 = [];
    for (var i = 0; i < arr.length; i++) {
        if (arr1.indexOf(arr[i]) === -1) {
            arr1.push(arr[i]);
        }
    }
    return arr1;
}
var arr = [1, 2, 3, 4, 3, 2, 5];
var arr1 = remDup(arr);
console.log('Original Array:', arr);
console.log('Array with Duplicates Removed:', arr1);
```

1. d. Binary Search

```
function bs(arr, t) {
    let l = 0;
    let r = arr.length - 1;
    while (l <= r) {
        const mid = Math.floor((l + r) / 2);
        if (arr[mid] === t) {
            return mid;
        } else if (arr[mid] < t) {
            l = mid + 1;
        } else {
            r = mid - 1;
        }
    }
    return -1;
}
const sortedArray = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
const targetValue = 7;
const result = bs(sortedArray, targetValue);
if (result !== -1) {
    console.log(`Element ${targetValue} found at index ${result}.`);
} else {
    console.log(`Element ${targetValue} not found in the array.`);
}
```

1. e. List the properties of a JavaScript object

```
let object = {
    name: 'Jack', age: 25, college: 'KMIT', year: 3, sem: 1
};
let properties = Object.keys(object)
console.log(properties);
```

1. f. to check whether an object contains given property

```
let object = {
    name: 'Jack', age: 25, college: 'KMIT', year: 3, sem: 1
};
console.log(object.hasOwnProperty('name'));
```

1. g. Quick sort

```
function quickSort(arr) {
    if (arr.length <= 1) {
        return arr;
    } else {
        const pivot = arr[arr.length - 1];
        const left = [];
        const right = [];
        for (let i = 0; i < arr.length - 1; i++) {
            if (arr[i] <= pivot) {
                left.push(arr[i]);
            } else {
                right.push(arr[i]);
            }
        }
        return [...quickSort(left), pivot, ...quickSort(right)];
    }
}
const unsortedArray = [3, 0, 2, 5, -1, 4, 1];
console.log("Original array:", unsortedArray);
const sortedArray = quickSort(unsortedArray);
console.log("Sorted array:", sortedArray);
```

1. h. Bubble Sort

```
function bubbleSort(arr) {
    const n = arr.length;
    for (let i = 0; i < n - 1; i++) {
        for (let j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                const temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    return arr;
}
const unsortedArray = [3, 0, 2, 5, -1, 4, 1];
console.log("Original array:", unsortedArray);
const sortedArray = bubbleSort(unsortedArray);
console.log("Sorted array:", sortedArray);
```