1. **Create a simple local module that will provide simple functions to add, subtract and multiplication and division. Use the module for different calculations.**

**Program:**

- **calc.js**

```
exports.add = function(a,b){
    return a+b;
}
exports.sub = function(a,b){
    return a-b;
}

exports.mul = function(a,b){
    return a*b;
}

exports.div = function(a,b){
    return a/b;
}
```

- **index.js**

```
const math = require('./calc');
var x = 10, y = 5
console.log("Addition:", math.add(x,y));
console.log("Subtraction:", math.sub(x,y));
console.log("Multiplication:", math.mul(x,y));
console.log("Division:", math.div(x,y));
```

**Output:**

Addition: 15

Subtraction: 5

Multiplication: 50

Division: 2

2. **Create a file, write few data in the file, append data, read the file synchronously and close the file.**

**Program:**

- **index.js**

```
const fs = require('fs');

const filePath = 'sample.txt';

// Write data to file
fs.writeFileSync(filePath, 'Hello, World!\n');

// Append data to file
fs.appendFileSync(filePath, 'Appending this text.\n');

// Read data from file
const data = fs.readFileSync(filePath, 'utf8');
console.log(data);
```

**Output:**

Hello, World!

Appended text.

3. **Write a redux based react JS application to concept of state using implement increment and decrement counter.**

**Step 1:** npx create-react-app counter

**Step 2:** npm install redux react-redux --save

**Step 3:** npm install @reduxjs/toolkit

**Step 4:** Create a folder "**store**" inside **src** folder and create a file **index.js** as

❖ **Program (src/store/index.js):**

```
import {configureStore, createSlice} from '@reduxjs/toolkit';

const counterSlice=createSlice({
    name: 'counter',
    initialState: { counter:0 },
    reducers:{
        increment(state,action){
            state.counter++;
        },
        decrement(state,action){
            state.counter--;
        }
    }
})

export const actions=counterSlice.actions;

const store=configureStore({
    reducer: counterSlice.reducer
})
export default store;
```

**Step 5: Inside the *src* folder modify the *App.js* file as:**

❖ **Program (src/App.js):**

```
import './App.css';
import { useSelector,useDispatch } from 'react-redux';
import { actions } from './store/index.js';

function App() {
    const counter =useSelector((state)=>state.counter);
    const dispatch=useDispatch();
    const increment=()=>{
        dispatch(actions.increment());
    };
    const decrement=()=>{
        dispatch(actions.decrement());
    };
  return (
    <div class="container">
      <h1>The counter test</h1>
    <h2>Counter:{ counter }</h2>
    <button onClick={increment}>Increment</button>
    <button onClick={decrement}>Decrement</button>
    </div>
  );
}
export default App;
```

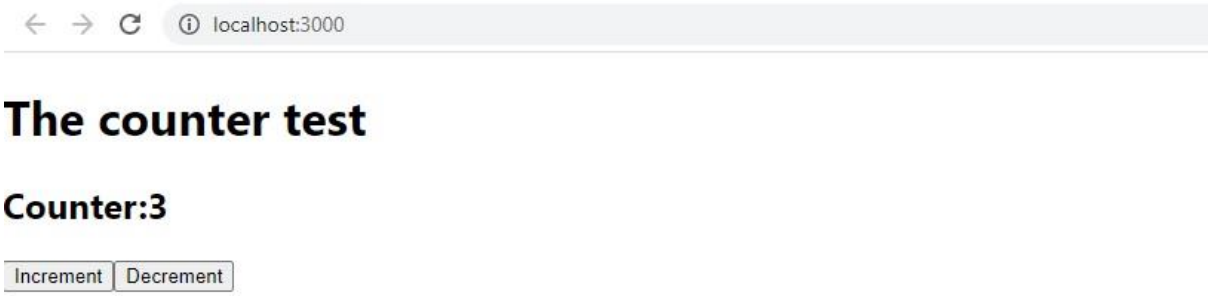
**Step 6: Inside the *src* folder modify the *index.js* file as:**

❖ **Program (src/index.js):**

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { Provider } from 'react-redux';
import store from './store/index.js';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
   <Provider store={store}>
  <React.StrictMode>
    <App />
  </React.StrictMode>
</Provider>
);
reportWebVitals();
```

**Step 7: npm start**

❖ **Output**




4. **Implement redux to check whether user is logged in or not.**

**Step 1: npx create-react-app counter**

**Step 2: npm install redux react-redux --save**

**Step 3: npm install @reduxjs/toolkit**

**Step 4: Create a folder "store" inside src folder and create a file index.js as**

❖ **Program (src/store/index.js):**

```
import {configureStore, createSlice} from '@reduxjs/toolkit';

const loginSlice=createSlice({
    name: 'login',
    initialState: { isLoggedIn:false },
    reducers:{
        sub_login(state,action){
            state.isLoggedIn=true;
        },
        sub_logout(state,action){
            state.isLoggedIn=false;
        }
    }
})

export const actions=loginSlice.actions;

const store=configureStore({
    reducer: loginSlice.reducer
})
export default store;
```

**Step 5: Inside the *src* folder modify the *App.js* file as:**

❖ **Program (src/App.js):**

```
import { useSelector, useDispatch } from 'react-redux';
import { store, actions } from './store/index.js';
import './App.css';

function checkLogin(islogin) {
  if (islogin)
    return "LoggedIn";
  else
    return "LoggedOut";
}

function App() {
  const islogin = useSelector((state) => state.isLoggedIn);
  const dispatch = useDispatch();
  const login_call = () => {
    dispatch(actions.sub_login());
  };
  const logout_call = () => {
    dispatch(actions.sub_logout());
  };
  console.log(islogin);

  return (
    <div>
      <h1>Log In form</h1>
      User Name: <input type="text" id="txtName" />
      <button onClick={login_call}>Login</button>
      <button onClick={logout_call}>Logout</button>
      <h2>User is {checkLogin(islogin)}</h2>
    </div>
  );
}

export default App;
```

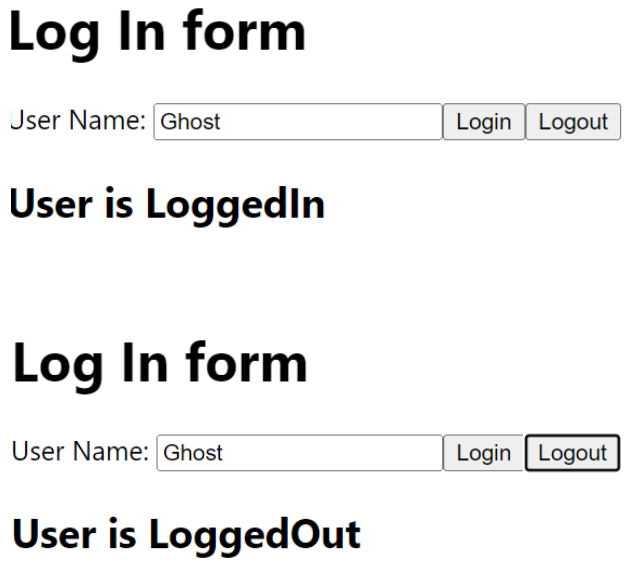
**Step 6: Inside the *src* folder modify the *index.js* file as:**

❖ **Program (src/index.js):**

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import {Provider} from 'react-redux';
import store from './store/index';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
<Provider store={store}>
  <React.StrictMode>
    <App />
  </React.StrictMode>
</Provider>
);

reportWebVitals();
```

**Step 7: npm start**

❖ **Output**



5. **Create a USER Module. Use this USER module to check logging credential of a user.**

**Step 1:** npm install prompt-sync

**Step 2:** npm install mongoose

**Step 3:** npm install express

**Step 4:** npm install –g @angular/cli

**Program:**

- **user.js**

```
var user1 = {
    first_name: "John",
    last_name: "Smith",
    age: "38",
    department: "Software",
    user: "john123",
    psw: "12jh23"
};
var user2 = {
    first_name: "Rohan",
    last_name: "Roy",
    age: "40",
    department: "Software",
    user: "rohan456",
    psw: "12rh78"
};

var users = new Array(user1, user2);

exports.log_check = function(user, psw) {
    for (let value of users) {
        if (user == value["user"] && psw == value["psw"])
            return true;
    }
    return false;
}
```

**Program:**

- **login_check.js**

```
const prompt = require('prompt-sync')();

const name = prompt('User name?');
const psw=prompt("Password?")

var log_mod=require("./user")

if(log_mod.log_check(name,psw)){
    console.log("Valid User");
}
else{
    console.log("Error:Invalid user.")
}
```

**Run:** ***node login_check.js***

**Output:**

User name?john123

Password?12jh23

Valid User

**Output 2:**

User name?hello

Password?123231

Error:Invalid user.

**6. Create date module which should return current date in " YYYY-MM-DD" format.**

**Program:**

- **date_mod.js**

```
exports.today = function () {

    let ts = Date.now();
    let date_ob = new Date(ts);

    let date = date_ob.getDate();
    let month = date_ob.getMonth() + 1;
    let year = date_ob.getFullYear();

    var dt = year + "-" + month + "-" + date;
    return dt;
}
```

- **test_date.js**

```
var current_date = require('./date_mod.js')

console.log("Today date is :" + current_date.today());
```

**Run:** ***node test_date.js***

**Output:**

Today date is :2024-6-2

7. **Create a file, write few data in the file, append data, read the file asynchronously and close the file.**

**Program:**

- **index.js**

```
const fs = require('fs');

const filePath = 'async-file.txt';
// Function to create a file asynchronously
function createFile() {
    fs.writeFile(filePath, 'Hello, World!\n', (err) => {
        if (err) throw err;
        console.log('File created successfully');
    });
}

// Function to append data to a file asynchronously
function appendToFile(data) {
    fs.appendFile(filePath, data, (err) => {
        if (err) throw err;
        console.log('Data appended successfully');
    });
}
// Function to read a file asynchronously
function readFile() {
    fs.readFile(filePath, 'utf8', (err, data) => {
        if (err) throw err;
        console.log('File content:');
        console.log(data);
    });
}

// Call functions sequentially
createFile();
appendToFile('Appended text.\n');
readFile();
```

**Run:** ***node index.js***

**Output:**

File created successfully

Data appended successfully

File content:

Hello, World!

8. **Create a file and write “Mallareddy University” and append ” AIML Department” to the same file.**

**Program:**

- **index.js**

```
var fs = require("fs");

// Write data to file
fs.writeFile('async-file.txt', 'Mallareddy University', function(err) {
    if (err) {
        return console.error(err);
    }
    console.log("Data written successfully!");
});

// Append data to file
fs.appendFile('input.txt', ' AIML Department', 'utf8',

    // Callback function
    function(err) {
        if (err) throw err;

        //  If no error
        console.log("Data is appended to file successfully.");
});
```

**Run:** ***node index.js***

**Output:**

Data written successfully!

Data is appended to file successfully.

**9. Create a HTTP server which will forward a HTML page in response of a client request.**

**Program:**

- **index.html**

```
<!DOCTYPE html>
<html>

<head>
    <title>My Website</title>
    <style>
      body{
        background-color: orange;
        color: white;
        font-size: 50px;
        text-align: center;
        margin-top: 15%;
        margin-bottom: 10%;
      }
    </style>
</head>

<body>
    <div class="center">
        <h1>Hello Again!</h1>
        <p>This is served from a file</p>
    </div>
</body>

</html>
```

**Program:**

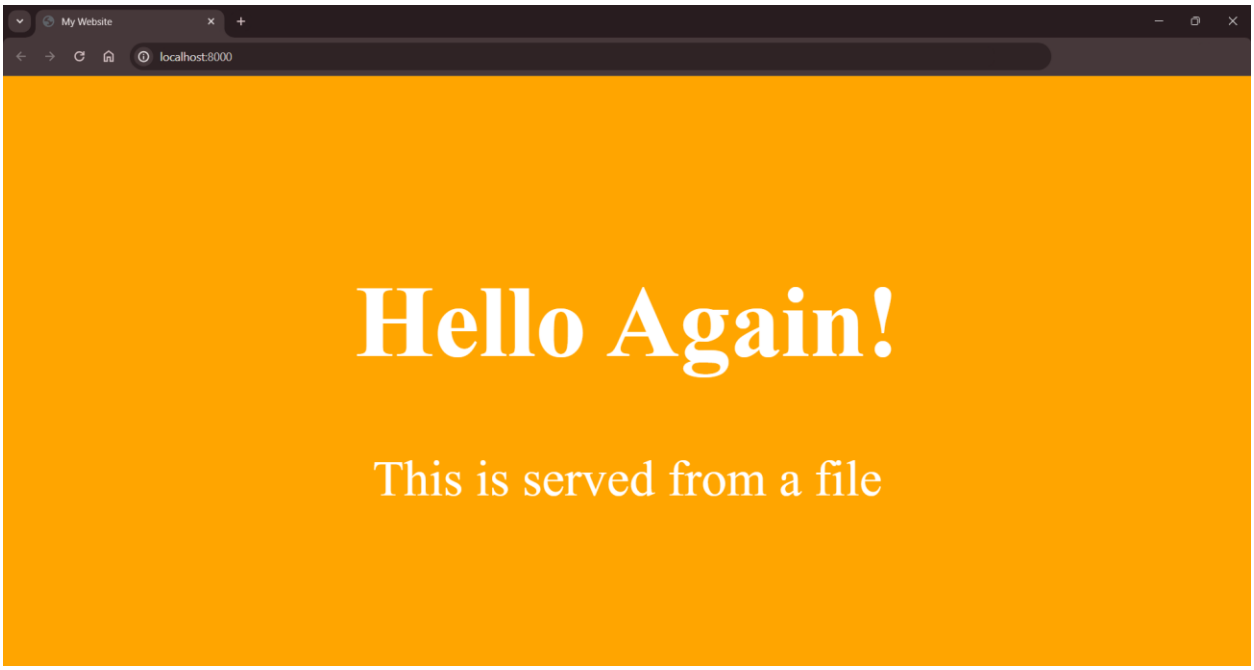
- **http_server.js**

```
const http = require("http");
const fs = require('fs').promises;

const host = 'localhost';
const port = 8000;

const requestListener = function (req, res) {
    fs.readFile(__dirname + "/index.html")
        .then(contents => {
            res.setHeader("Content-Type", "text/html");
```

```
            res.writeHead(200);
            res.end(contents);
        })
        .catch(err => {
            res.writeHead(500);
            res.end(err);
            return;
        });
};

const server = http.createServer(requestListener);
server.listen(port, host, () => {
    console.log(`Server is running on http://${host}:${port}`);
});
```

**Run:** ***node http_server.js***

**Output:**



**10. Create a HTTP server which will respond by sending message “server is active”**

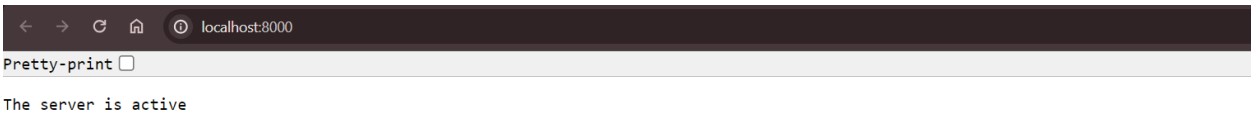
**Program:**

- **http_new_server.js**

```
const http = require("http");
const fs = require('fs').promises;
const host = 'localhost';
const port = 8000;

const requestListener = function (req, res) {
    res.setHeader("Content-Type", "application/json");
    res.writeHead(200);
    res.end("The server is active");
}
const server = http.createServer(requestListener);
server.listen(port, host, () => {
    console.log(`Server is running on http://${host}:${port}`);
});
```

**Run:** ***node http_new_server.js***

**Output:**



**11. Create database - “employee”. Create a collection in the employee database and insert the following data. Update salary of “Smith”. Delete the record for “Binod”. Drop collection and finally drop the database.**

| Name | address | Phone_no | Salary |
|---|---|---|---|
| Binod | Hyderabad | 9000789211 | 10000 |
| Smith | Delhi | 9825800212 | 15000 |
| John | Mumbai | 9725652582 | 8000 |

**Step 1: Create Database**

- **Syntax:**
  use employee
- **Output:**
  *switched to db employee*

**Step 2: Create Collection and Insert Data**

- **Syntax:**

```
db.employee.insertMany([
    { "Name": "Binod", "address": "Hyderabad", "Phone_no": 9000789211,
"Salary": 10000 },
    { "Name": "Smith", "address": "Delhi", "Phone_no": 9825800212,
"Salary": 15000 },
    { "Name": "John", "address": "Mumbai", "Phone_no": 9725652582,
"Salary": 8000 }
  ])
```

- **Output:**

```
{
  "acknowledged" : true,
  "insertedIds" : [
     ObjectId("60b7e6d054fc3f36c3a3d312"),
     ObjectId("60b7e6d054fc3f36c3a3d313"),
     ObjectId("60b7e6d054fc3f36c3a3d314")
  ]
}
```

**Step 3: Update Salary of "Smith"**

- **Syntax:**

```
db.employee.updateOne(
    { "Name": "Smith" },
    { $set: { "Salary": 20000 } }
  )
```

- **Output**:
  *{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }*

**Step 4: Delete Record for "Binod"**

- **Syntax:**
  db.employee.deleteOne({ "Name": "Binod" })
- **Output:**
  *{ "acknowledged" : true, "deletedCount" : 1 }*

**Step 5: Drop Collection**

- **Syntax:**
  db.employee.drop()
- **Output:**
  *{ "ok" : 1 }*

**Step 6: Drop Database**

- **Syntax:**
  use employee
  db.dropDatabase()
- **Output:**
  *{ "dropped" : "employee", "ok" : 1 }*

**12. Create JSON file data.json with following Table data and import the data into the mongodb database. After importing display the data in command shell, add 2 marks to each student, change phone no for Binod to 9101884569.**

| Student_Name | Student_address | Phone_no | Marks |
|---|---|---|---|
| **Binod** | **Hyderabad** | **9000790211** | **88** |
| **Smith** | **Delhi** | **9825801112** | **90** |
| **John** | **Mumbai** | **9725772500** | **65** |

**Step 1: Create a JSON file named `data.json` with the provided table data.**

```
 [
  {
    "name": "Alice",
    "age": 20,
    "marks": 85,
    "phone": "1234567890"
  },
  {
    "name": "Bob",
    "age": 21,
    "marks": 75,
    "phone": "9876543210"
  },
  {
    "name": "Binod",
    "age": 22,
    "marks": 80,
    "phone": "9876543211"
  }
]
```

**Step 2:** Open your MongoDB client command line interface.

**Step 3:** Create a database:

```
use student
```

**Step 4:** Create collections:

```
db.createCollection("students_details")
```

**Step 5:** Import the data from the JSON file into MongoDB.

```
mongoimport --db student --collection students --file data.json
```

**Step 6:** Display the imported data in the MongoDB shell.

```
use student
db.students.find()
```

**Step 7:** Add 2 marks to each student and change the phone number for Binod.

```
db.students.updateMany({}, {$inc: {marks: 2}})
db.students.updateOne({name: "Binod"}, {$set: {phone: "9101884569"}})
```

**Step 8:** Display the updated data in the MongoDB shell.

```
db.students.find()
```