Week3: Introduction to Node.js and its Modules

Topics to Cover:

- What is Node.js?
- Installing Node.js.
- Using the Node.js REPL.
- Core modules (fs, path, os).
- Writing and running a simple script.

Project: System Utility Dashboard

- Objective: Create a node application that provides system information.

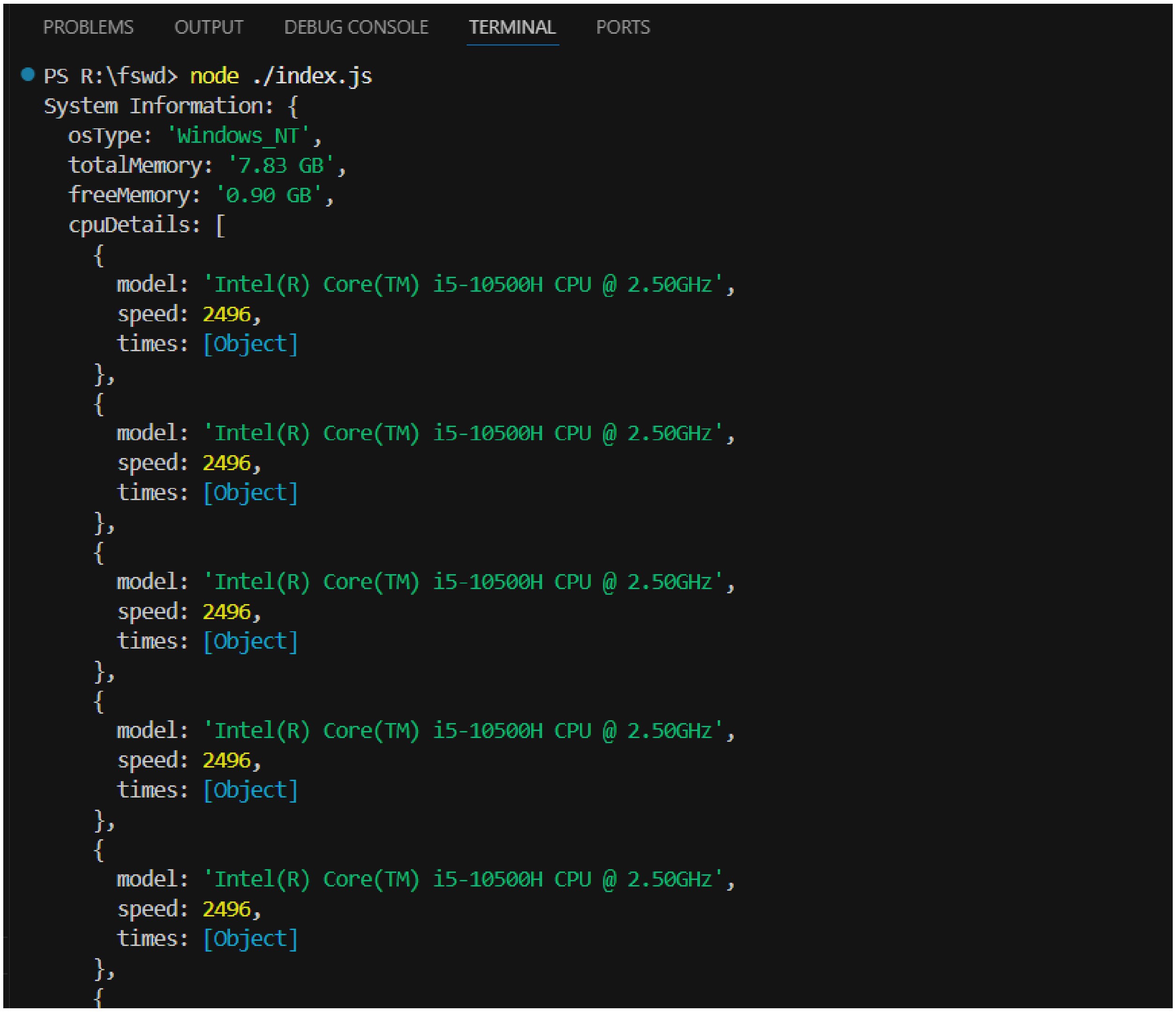
Tasks:

- Use the os module to display the OS type, total memory, free memory, and CPU details.
- Use the fs module to save the system information to a log file.
- Use the path module to ensure the file is saved in a standardized format (logs/system-info.txt).

Input:-
```
const os = require('os');
const fs = require('fs');
const path = require('path');
function getSystemInfo() {
return {
osType: os.type(),
totalMemory: `${(os.totalmem() / (1024 **
3)).toFixed(2)} GB`,
freeMemory: `${(os.freemem() / (1024 **
3)).toFixed(2)} GB`,
cpuDetails: os.cpus()
};
}
function saveSystemInfoToFile(info) {
const logDir = path.join(__dirname, 'logs');
const logFilePath = path.join(logDir,
'system-info.txt');
if (!fs.existsSync(logDir)) {
fs.mkdirSync(logDir);
}
const logData = `System Information:
OS Type: ${info.osType}
Total Memory: ${info.totalMemory}
```

```
Free Memory: ${info.freeMemory}
CPU Details: ${JSON.stringify(info.cpuDetails, null,
2)}
`;
fs.writeFileSync(logFilePath, logData, 'utf-8');
console.log(`System information saved to
${logFilePath}`);
}
const systemInfo = getSystemInfo();
console.log('System Information:', systemInfo);
saveSystemInfoToFile(systemInfo);
```

Output:-



```
PS R:\fswd> node ./index.js
System Information: {
  osType: 'Windows_NT',
  totalMemory: '7.83 GB',
  freeMemory: '0.90 GB',
  cpuDetails: [
    {
      model: 'Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz',
      speed: 2496,
      times: [Object]
    },
    {
      model: 'Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz',
      speed: 2496,
      times: [Object]
    },
    {
      model: 'Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz',
      speed: 2496,
      times: [Object]
    },
    {
      model: 'Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz',
      speed: 2496,
      times: [Object]
    },
    {
      model: 'Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz',
      speed: 2496,
      times: [Object]
    },
    {
```

```
10    3)).toFixed(2)} GB`,
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

      model: 'Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz',
      speed: 2496,
      times: [Object]
    },
    {
      model: 'Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz',
      speed: 2496,
      times: [Object]
    },
    {
      model: 'Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz',
      speed: 2496,
      times: [Object]
    },
    {
      model: 'Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz',
      speed: 2496,
      times: [Object]
    }
  ]
}
System information saved to
R:\fswd\logs\system-info.txt
PS R:\fswd>
```

Working with Custom Modules and HTTP

Topics to Cover:

- Custom modules: require and module.exports.
- Using the HTTP module to create a server.
- Handling different URL paths manually.

Project: Basic Static File Server

- Objective: Build an HTTP server to serve static files like HTML, CSS, and images.

Tasks:

- Create a simple HTML page with a welcome message and a few images.
- Use the http and fs modules to serve these files when the browser requests them.

- Add logic to handle 404 errors when a file is not found.

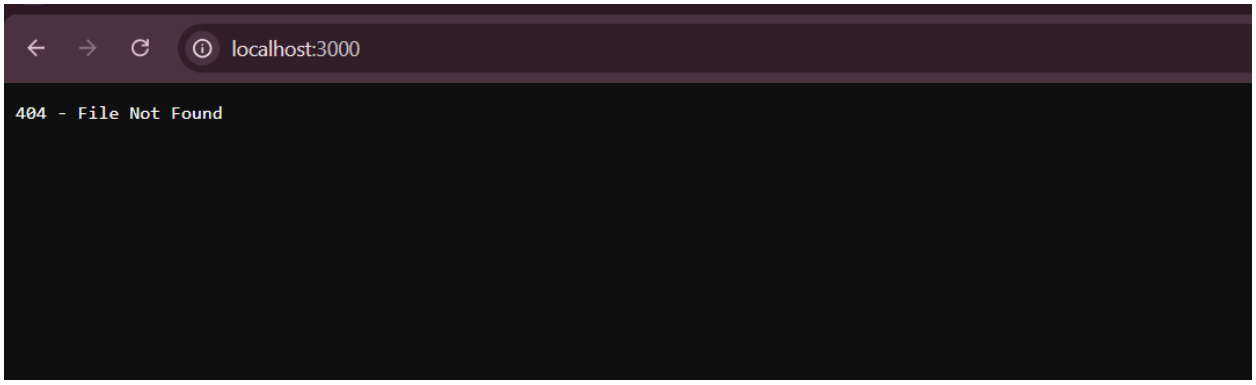
Input:-const http = require('http');

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

// Create the server
const server = http.createServer((req, res) => {
  // Check if the requested file is an image or other static file
  let filePath = path.join(__dirname, req.url === '/' ? 'image1.jpg' : req.url); // Default to 'image1.jpg'

  // Get the file extension and set the appropriate content type
  const extname = path.extname(filePath);
  let contentType = 'application/octet-stream'; // Default content type

  switch (extname) {
    case '.jpg':
    case '.jpeg':
      contentType = 'image/jpeg';
      break;
    case '.png':
      contentType = 'image/png';
      break;
    case '.gif':
      contentType = 'image/gif';
      break;
    default:
      contentType = 'application/octet-stream'; // Default for unsupported files
  }

  // Read the requested file from the filesystem
  fs.readFile(filePath, (err, content) => {
    if (err) {
      if (err.code === 'ENOENT') {
        // If file not found, send a 404 error response
        res.writeHead(404, { 'Content-Type': 'text/plain' });
        res.end('404 - File Not Found', 'utf-8');
      } else {
        // Server error
```

```
            res.writeHead(500);
            res.end('500 - Server Error: ' + err.code);
        }
    } else {
        // If file is found, serve it
        res.writeHead(200, { 'Content-Type': contentType });
        res.end(content, 'utf-8');
    }
  });
});

// Define the port the server will listen to
const port = 3000;

// Start the server
server.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

Output:-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS R:\fswd> node ./index.js
Server running at http://localhost:3000
```


localhost:3000
404 - File Not Found