

Practical 6

Link to my GitHub repository:

<https://github.com/RaviThakur322/Node-JS-Practical-GF2023485999/tree/main>

1. Write a program to create the server with the dynamic imports with top level await.
2. Goal: Use stream, readline, pipeline, backpressure.
Tasks: Given data/users.csv (~1M rows), count users per domain from email. Write results to out/domains.json without loading all into memory.
3. Create an EventEmitter-based logger with transports and rotation: Logger emits log events; transports subscribe (console, file). Implement size-based rotation at ~50KB per file.

1. logger.mjs - An EventEmitter-based logger with console and file transports, including size-based log rotation.

```
import http from 'http';
import { mkdir, writeFile } from 'fs/promises';

// Create necessary directories and a dummy data file for the demo
async function setup() {
  try {
    await mkdir('data', { recursive: true });
    await mkdir('out', { recursive: true });
    await mkdir('logs', { recursive: true });

    // Generate a small sample CSV for demonstration with Indian names
    const names = [
      'Priya Sharma', 'Rohan Gupta', 'Anjali Singh', 'Vikram Kumar', 'Sneha Patel',
      'Arjun Reddy', 'Meera Desai', 'Aditya Joshi', 'Pooja Mehta', 'Sameer Khan',
      'Kavita Rao', 'Rajesh Nair', 'Sunita Murthy', 'Deepak Iyer', 'Lakshmi Pillai'
    ];
    const csvContent = Array.from({ length: 5000 }, (_, i) => {
      const name = names[i % names.length];
      const emailName = name.toLowerCase().replace(' ', '.');
      const domain = ['example.com', 'test.org', 'sample.net', 'demo.co.uk', 'mail.io'][i % 5];
      return `${i + 1},${name},${emailName}.${i}@${domain}`;
    }).join('\n');
    await writeFile('data/users.csv', 'id,name,email\n' + csvContent);
    console.log('Setup complete. Dummy data generated.');
```

```
  } catch (err) {
    console.error('Error during setup:', err);
  }
}

// Run setup before starting the server
await setup();

const PORT = 3000;

const server = http.createServer(async (req, res) => {
  const { url, method } = req;
  console.log(`Received ${method} request for ${url}`);

  res.setHeader('Content-Type', 'application/json');

  // Simple routing
  switch (url) {
    case '/':
      res.writeHead(200);
      res.end(JSON.stringify({ message: 'Server is running. Visit /log or /process to trigger actions.' }));
```

2.process-csv.mjs - A script to efficiently process a large CSV file using streams, handling backpressure.

```
import express from 'express';
import { mkdir, writeFile } from 'fs/promises';

// Create necessary directories and a dummy data file for the demo
async function setup() {
  try {
    await mkdir('data', { recursive: true });
    await mkdir('out', { recursive: true });
    await mkdir('logs', { recursive: true });

    // Generate a small sample CSV for demonstration with Indian names
    const indianNames = [
      'Priya Sharma', 'Rohan Gupta', 'Anjali Singh', 'Vikram Kumar', 'Sneha Patel',
      'Arjun Reddy', 'Meera Desai', 'Aditya Joshi', 'Pooja Mehta', 'Sameer Khan',
      'Kavita Rao', 'Rajesh Nair', 'Sunita Murthy', 'Deepak Iyer', 'Lakshmi Pillai'
    ];
    const csvContent = Array.from({ length: 5000 }, (_, i) => {
      const name = indianNames[i % indianNames.length];
      const emailName = name.toLowerCase().replace(' ', '.');
      const domain = ['example.com', 'test.org', 'sample.net', 'demo.co.uk', 'mail.io'][i % 5];
      return `${i + 1},${name},${emailName}.${i}@${domain}`;
    }).join('\n');
    await writeFile('data/users.csv', 'id,name,email\n' + csvContent);
    console.log('Setup complete. Dummy data generated.');
```

```
} catch (err) {
  console.error('Error during setup:', err);
}

// Run setup before starting the server
await setup();

const PORT = 3000;

const server = http.createServer(async (req, res) => {
  const { url, method } = req;
  console.log(`Received ${method} request for ${url}`);

  res.setHeader('Content-Type', 'application/json');

  // Simple routing
  switch (url) {
    case '/':
      res.writeHead(200);
      res.end(JSON.stringify({ message: 'Server is running. Visit /log or /process to trigger actions.' }));
```

3. server.mjs - An HTTP server that uses dynamic imports and top-level await to run the other modules on demand.

```
JS event_emitter_example.js JS index.js JS logger.mjs Extension: Rainbow CSV JS process-csv.mjs readme.txt JS server.mjs X users.csv

Practical 6 > JS server.mjs > setup
1 import http from 'http';
2 import { mkdir, writeFile } from 'fs/promises';
3
4 // Create necessary directories and a dummy data file for the demo
5 async function setup() {
6   try {
7     await mkdir('data', { recursive: true });
8     await mkdir('out', { recursive: true });
9     await mkdir('logs', { recursive: true });
10
11     // Generate a small sample CSV for demonstration with Indian names
12     const indianNames = [
13       'Priya Sharma', 'Rohan Gupta', 'Anjali Singh', 'Vikram Kumar', 'Sneha Patel',
14       'Arjun Reddy', 'Meera Desai', 'Aditya Joshi', 'Pooja Mehta', 'Sameer Khan',
15       'Kavita Rao', 'Rajesh Nair', 'Sunita Murthy', 'Deepak Iyer', 'Lakshmi Pillai'
16     ];
17     const csvContent = Array.from({ length: 5000 }, (_, i) => {
18       const name = indianNames[i % indianNames.length];
19       const emailName = name.toLowerCase().replace(' ', '.');
20       const domain = ['example.com', 'test.org', 'sample.net', 'demo.co.uk', 'mail.io'][i % 5];
21       return `${i + 1},${name},${emailName}.${i}@${domain}`;
22     }).join('\n');
23     await writeFile('data/users.csv', 'id,name,email\n' + csvContent);
24     console.log('Setup complete. Dummy data generated.');
```

```
25   } catch (err) {
26     console.error('Error during setup:', err);
27   }
28 }
29
30 // Run setup before starting the server
31 await setup();
32
33 const PORT = 3000;
34
35 const server = http.createServer(async (req, res) => {
36   const { url, method } = req;
37   console.log(`Received ${method} request for ${url}`);
38
39   res.setHeader('Content-Type', 'application/json');
40
41   // Simple routing
42   switch (url) {
43     case '/':
44       res.writeHead(200);
45       res.end(JSON.stringify({ message: 'Server is running. Visit /log or /process to trigger actions.' }));
46   }
```

Output:

```
Node.js v24.8.0
ravithakur@Ravis-MacBook-Air Node-JS-Practical-GF2023485999 % node "/Users/ravithakur/Documents/GitHub/Node/Node-JS-Practica
Setup complete. Dummy data generated.
🔥 Server listening on http://localhost:3000
Endpoints:
  - /      : Home
  - /log   : Trigger the logger
  - /process : Trigger the CSV stream processing
Received GET request for /process
Starting CSV processing...
✅ CSV processing complete. Results saved to out/domains.json
```

```
localhost:3000/process

New Tab Learn Self-Confid... How to Find the Ri... AI Adobe Acrobat Ho... Big Data Analytics:... Adobe Acrobat

Pretty-print
{"message":"CSV processing initiated and completed.","result":{"totalDomains":6,"topDomain":["example.com",1000]}}
```

```
localhost:3000

New Tab Learn Self-Confid... How to Find the Ri... AI Adobe Acrobat Ho... Big Data Analytics:...

Pretty-print
{"message":"Server is running. Visit /log or /process to trigger actions."}
```