Practical 4

Project: File Organizer Tool

Objective: Build a CLI tool to organize files into folders based on their type (e.g., images, documents, videos).

Tasks:

* Accept a directory path as an input from the user.
* Use the fs module to read all files in the directory.
* Move files into folders like Images, Documents, and Others based on their

extensions.

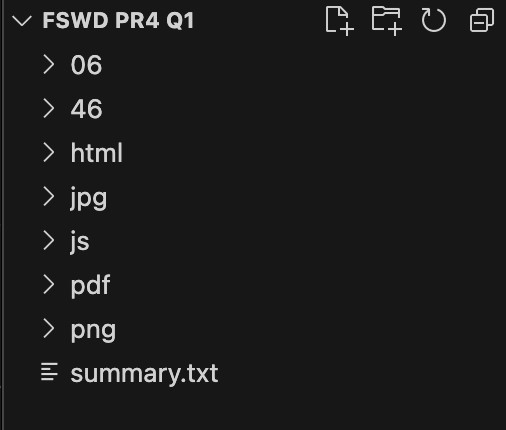
* Log the operations performed into a summary.txt file.

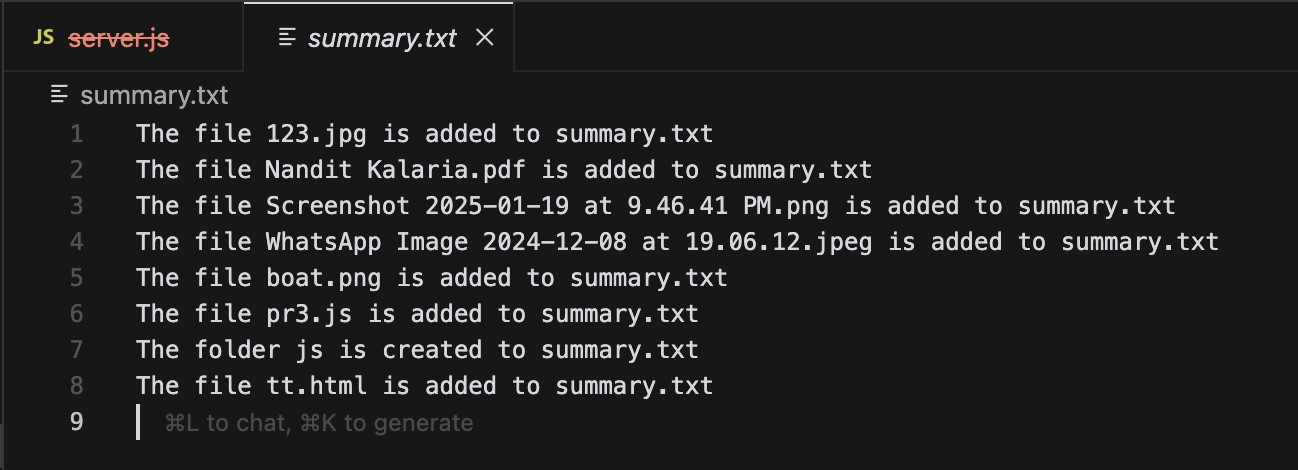
Screenshots of Code:

|  |
| --- |
| import fs from 'fs'; import fsp from 'fs/promises'; |
| import path from 'path'; |
| import { fileURLToPath } from "url"; import readline from 'readline'; |
|  |
| const r1 = readline.createInterface({ |
| input: process.stdin, output: process.stdout |
| }); |
| r1.question('Please enter the directory path: ', async(*directoryPath*) => { |
| let directoryFiles = await fsp.readdir(*directoryPath*); |
| for (const fileName of directoryFiles) { |
| let fileParts = fileName.split("."); if (fileParts.length > 1) { |
| let fileExtension = fileParts[1]; |
| if (fs.existsSync(path.join(*directoryPath*, fileExtension))) { fsp.rename(path.join(*directoryPath*, fileName), path.join(*directoryPath*, |
| fileExtension, fileName)); |
| fs.appendFileSync('summary.txt', `The folder ${fileExtension} is |
| created to summary.txt\n`) } else { |
| fs.mkdirSync(path.join(*directoryPath*, fileExtension)); |
| fsp.rename(path.join(*directoryPath*, fileName), path.join(*directoryPath*, fileExtension, fileName)); |
| fs.appendFileSync('summary.txt', `The file ${fileName} is added to |
| summary.txt\n`) } |
| } |
| } |
| r1.close(); |
| }) |

Screenshots of Output:







Project: User Management System

Objective: Create a RESTful API to manage user data without using Express.js.

Tasks:

* Implement the following endpoints using the http module:
* GET /users: Return a list of all users stored in a JSON file.
* POST /users: Accept new user data in the request body and add it to the JSON file.
* DELETE /users/:id: Remove a user by their ID from the JSON file.
* Use the fs module to store and retrieve user data persistently.
* Test the API using Postman or curl.

Screenshots of Code:

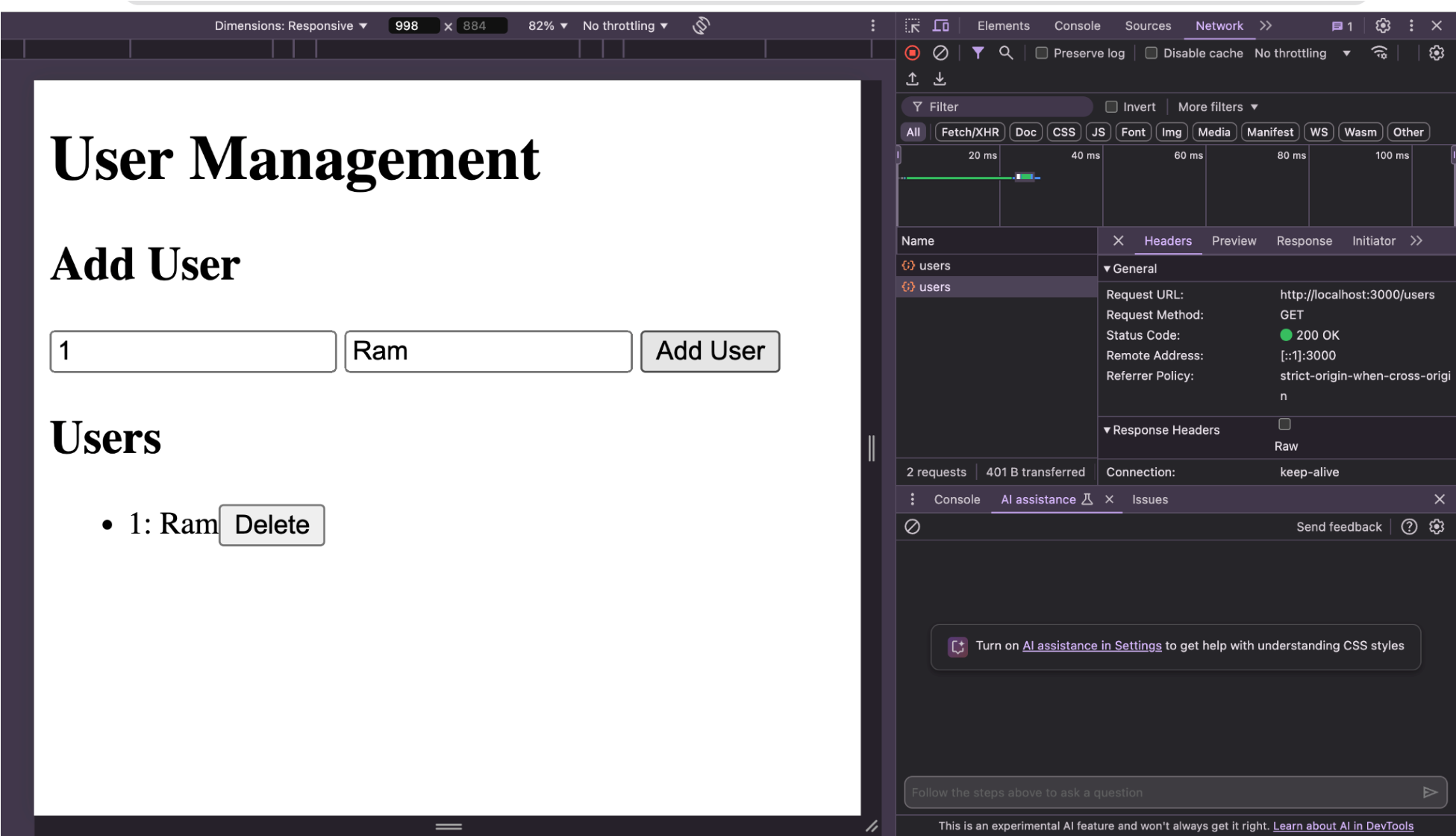
Server.js

|  |
| --- |
| const http = require('http'); |
| const fs = require('fs'); const url = require('url'); |
| const path = require('path'); |
|  |
| const PORT = 3000; const DATA\_FILE = 'users.json'; |
|  |

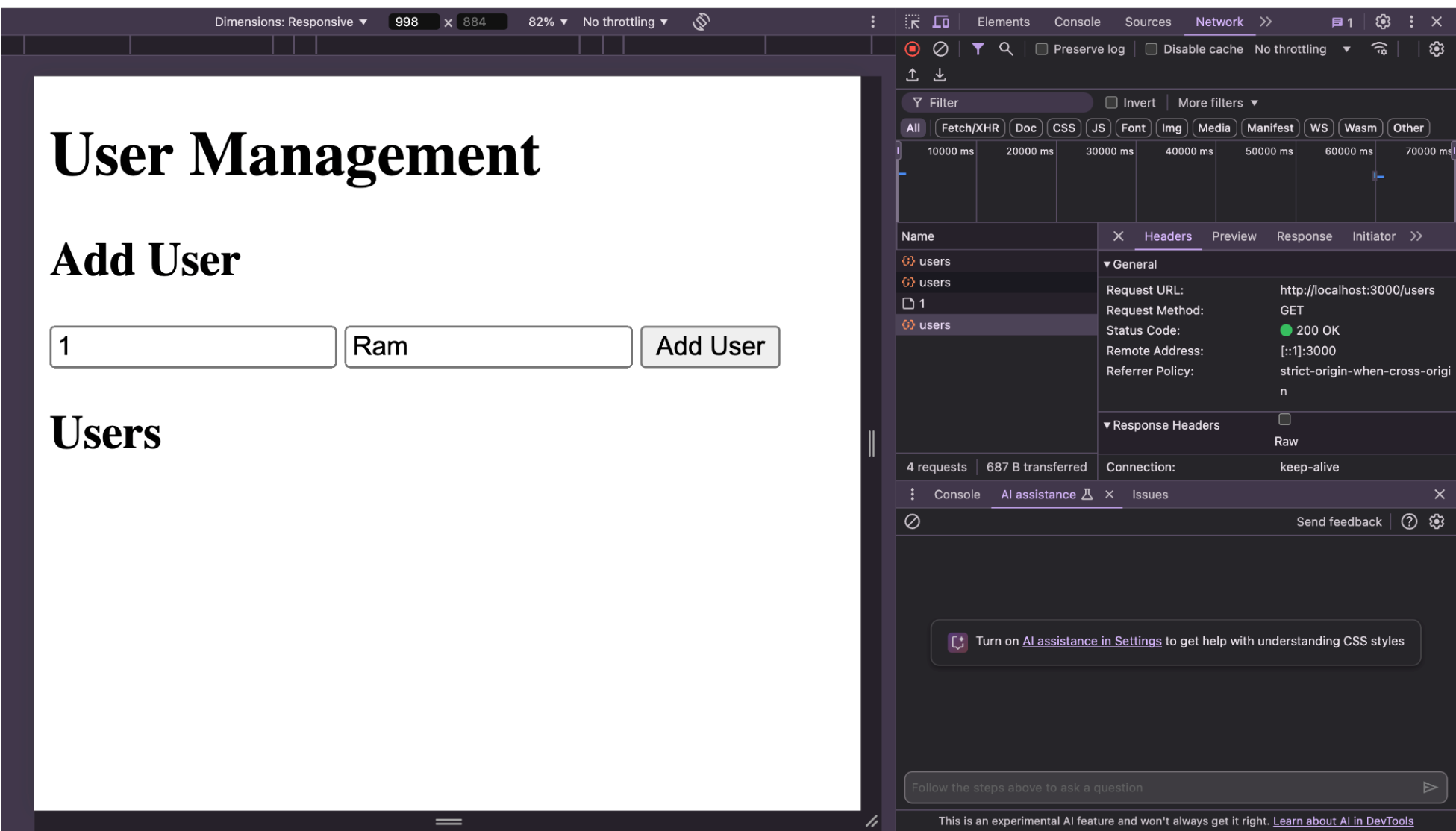
|  |
| --- |
| const readUsers = () => { |
| const data = fs.readFileSync(DATA\_FILE); return JSON.parse(data); |
| }; |
|  |
| const writeUsers = (*users*) => { fs.writeFileSync(DATA\_FILE, JSON.stringify(*users*, null, 2)); |
| }; |
| const server = http.createServer((*req*, *res*) => { |
| console.log(`Received ${*req*.method} request for ${*req*.url}`); |
| const parsedUrl = url.parse(*req*.url, true); const method = *req*.method; |
|  |
| if (parsedUrl.pathname === '/') { |
| fs.readFile(path.join(\_\_dirname, 'index.html'), (*err*, *data*) => { if (*err*) { |
| *res*.writeHead(500); |
| return *res*.end('Error loading index.html'); } |
| *res*.writeHead(200, { 'Content-Type': 'text/html' }); |
| *res*.end(*data*); |
| }); return; |
| } |
| if (method === 'GET' && parsedUrl.pathname === '/users') { |
| const users = readUsers(); |

|  |
| --- |
| *res*.writeHead(200, { 'Content-Type': 'application/json' }); |
| *res*.end(JSON.stringify(users)); } else if (method === 'POST' && parsedUrl.pathname === '/users') { |
| let body = ''; |
| *req*.on('data', *chunk* => { |
| body += *chunk*.toString(); }); |
| *req*.on('end', () => { |
| const newUser = JSON.parse(body); const users = readUsers(); |
| users.push(newUser); |
| writeUsers(users); *res*.writeHead(201, { 'Content-Type': 'application/json' }); |
| *res*.end(JSON.stringify(newUser)); |
| }); |
| } else if (method === 'DELETE' && parsedUrl.pathname.startsWith('/users/')) { const id = parsedUrl.pathname.split('/')[2]; |
| let users = readUsers(); |
| users = users.filter(*user* => *user*.id !== id); writeUsers(users); |
| *res*.writeHead(204); |
| *res*.end(); |
| } else { *res*.writeHead(404, { 'Content-Type': 'text/plain' }); |
| *res*.end('Not Found'); |
| }  }); |
|  |
| server.listen(PORT, () => { |
| console.log(`Server is running on http://localhost:${PORT}`); }); |
|  |

Screenshots of Output:



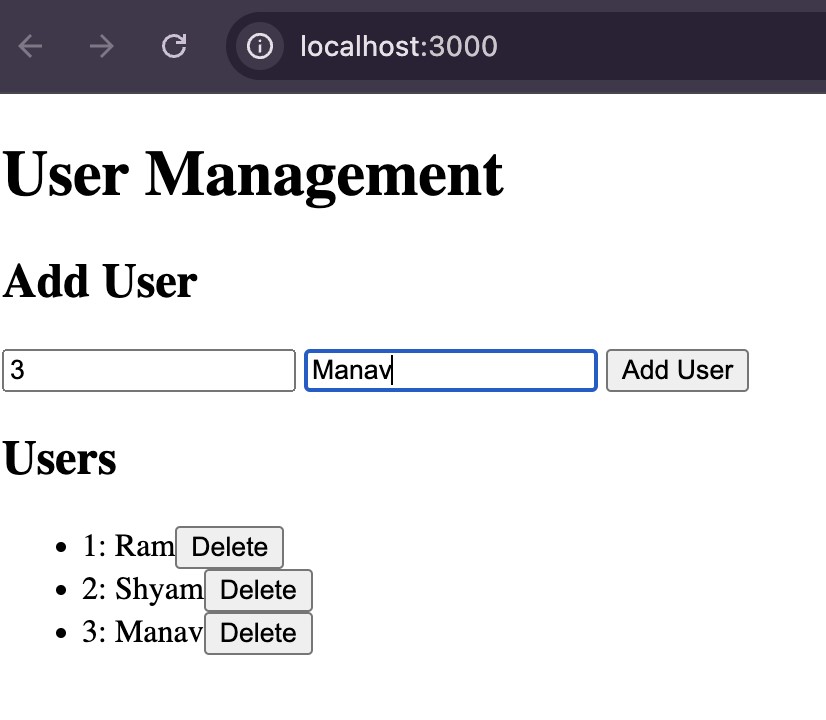


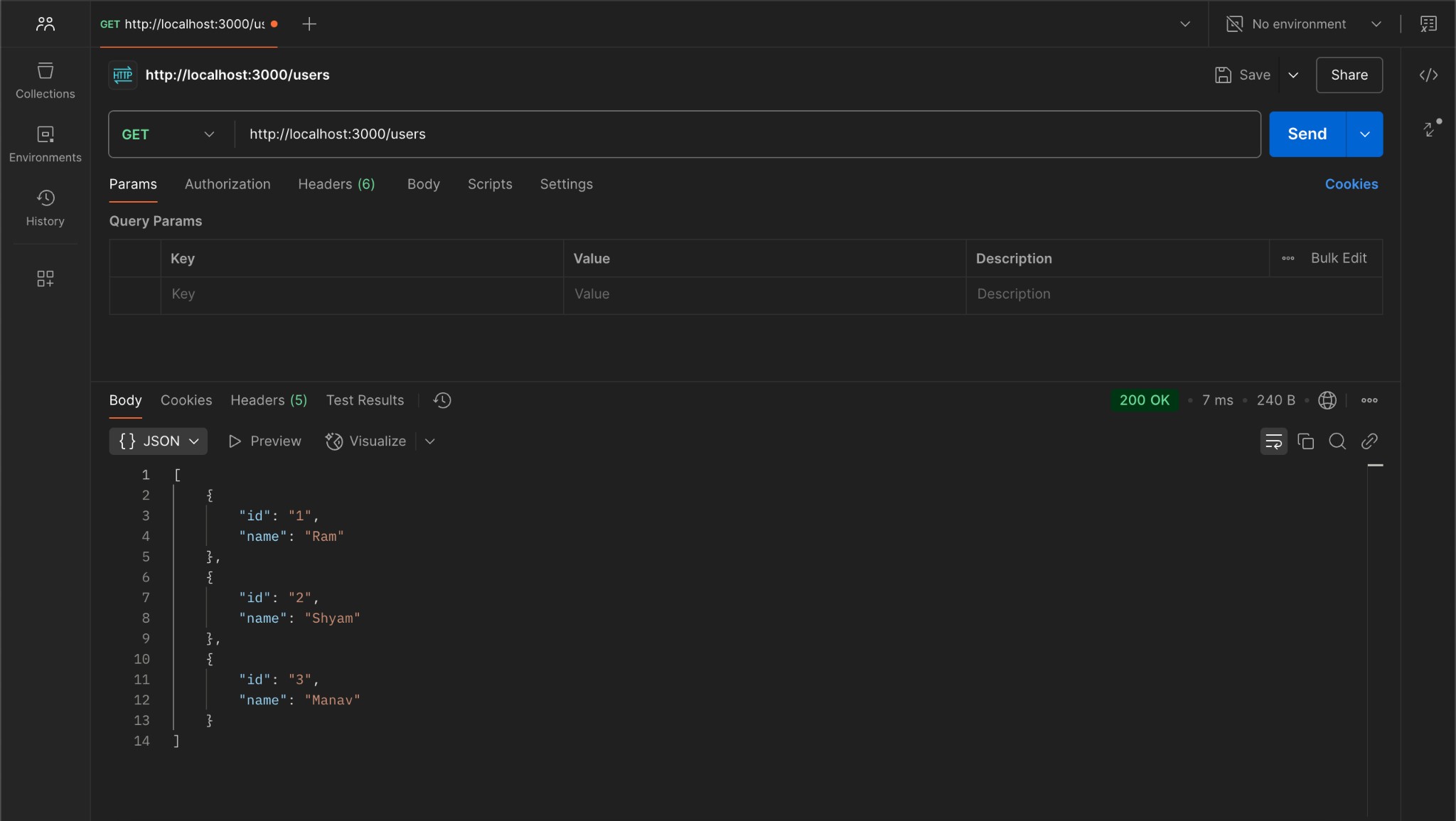




Postman

Testing:





Conclusion:

I learned how to use the fs module in Node.js for file manipulation and how to categorize files based on their extensions. Additionally, I gained experience in logging operations to a text file for tracking purposes. I learned to utilize the built-in http module for handling requests and responses, as well as the fs module for persistent data storage in JSON format. This project also enhanced my skills in API testing using Postman.