# TỔNG HỢP DỰ ÁN - MÔ TẢ TOÀN BỘ CODE

## 1. server.js

File chính khởi tạo server HTTP/HTTPS và WebSocket, xử lý API, phục vụ file tĩnh, quản lý lỗi.

- Sử dụng Express.js để tạo server.

- Middleware gán Request-Id, custom headers cho mọi request.

- Phục vụ file tĩnh từ thư mục `public/` (HTML, CSS, JS), cho phép cache.

- API endpoint `/api/info`: trả về thông tin hệ thống (timestamp, uptime, version, platform, memory, client headers).

- API endpoint `/api/boom`: trả về lỗi 500 để kiểm thử xử lý lỗi.

- Xử lý 404 cho các route không tồn tại, trả về HTML, JSON hoặc text.

- Global error handler cho lỗi 500.

- Khởi tạo server HTTP hoặc HTTPS (nếu có chứng chỉ trong `certs/`).

- Khởi tạo WebSocket server (port 3001), gửi/nhận tin nhắn realtime.

## 2. client.js

Xây dựng HTTP client từ đầu, không dùng axios/fetch, hỗ trợ GET/POST cho cả HTTP và HTTPS.

- Class `SimpleHttpClient` với các phương thức:

- `request(method, url, options)`: gửi request, nhận response, trả về status, headers, body.

- `get(url, headers)`: gửi GET request.

- `post(url, body, headers)`: gửi POST request, tự động chuyển object thành JSON.

- Hàm kiểm thử `runTests()`:

- GET tới server nội bộ `/api/info`.

- GET tới GitHub API.

- POST tới JSONPlaceholder.

- Xử lý lỗi khi server không khả dụng.

- Log chi tiết request/response ra console.

## 3. tcp-server.js

Khởi tạo TCP server sử dụng Node.js module `net`.

- Lắng nghe port 4000.

- Khi client kết nối, gửi thông báo chào mừng.

- Khi nhận dữ liệu từ client, phản hồi lại với nội dung nhận được.

## 4. tcp-client.js

TCP client kết nối tới TCP server.

- Kết nối tới port 4000.

- Khi kết nối thành công, gửi chuỗi “Xin chào server!”.

- Nhận dữ liệu từ server, in ra console.

- Xử lý ngắt kết nối.

## 5. ws-demo.js

Demo WebSocket server và client.

- WebSocket server chạy port 3002, gửi thông báo chào mừng khi client kết nối, phản hồi lại tin nhắn nhận được.

- WebSocket client kết nối tới server, gửi tin nhắn “Xin chào server!”, nhận và in ra phản hồi từ server.

## 6. benchmark.js

Script đo hiệu năng tải file tĩnh với các kích thước khác nhau.

- Danh sách file: `public/style.css`, `public/script.js`, `public/index.html`.

- Hàm `benchmark(file)`: gửi HTTP GET tới từng file, đo thời gian tải và kích thước file, in ra console.

## 7. monitor.js

Tiện ích giám sát mạng, log chi tiết các loại request.

- Hàm `logRequest(url, method)`: gửi request tới URL, đo thời gian phản hồi, phân loại request (static/dynamic), log status, size, thời gian.

- Hàm `runMonitor()`: kiểm thử các loại request (static, dynamic, 404).

## 8. public/

Giao diện web demo tương tác với server.

- `index.html`: Trang chủ, có nút bấm lấy dữ liệu server qua AJAX, hiển thị kết quả.

- `script.js`: Xử lý logic gửi request từ web, nhận kết quả từ server.

- `style.css`: Giao diện responsive, hiện đại.

## 9. certs/

Chứa chứng chỉ SSL/TLS để chạy HTTPS server.

- `server.key`, `server.crt`: dùng cho HTTPS.

## 10. docs/, presentation/, screenshots/

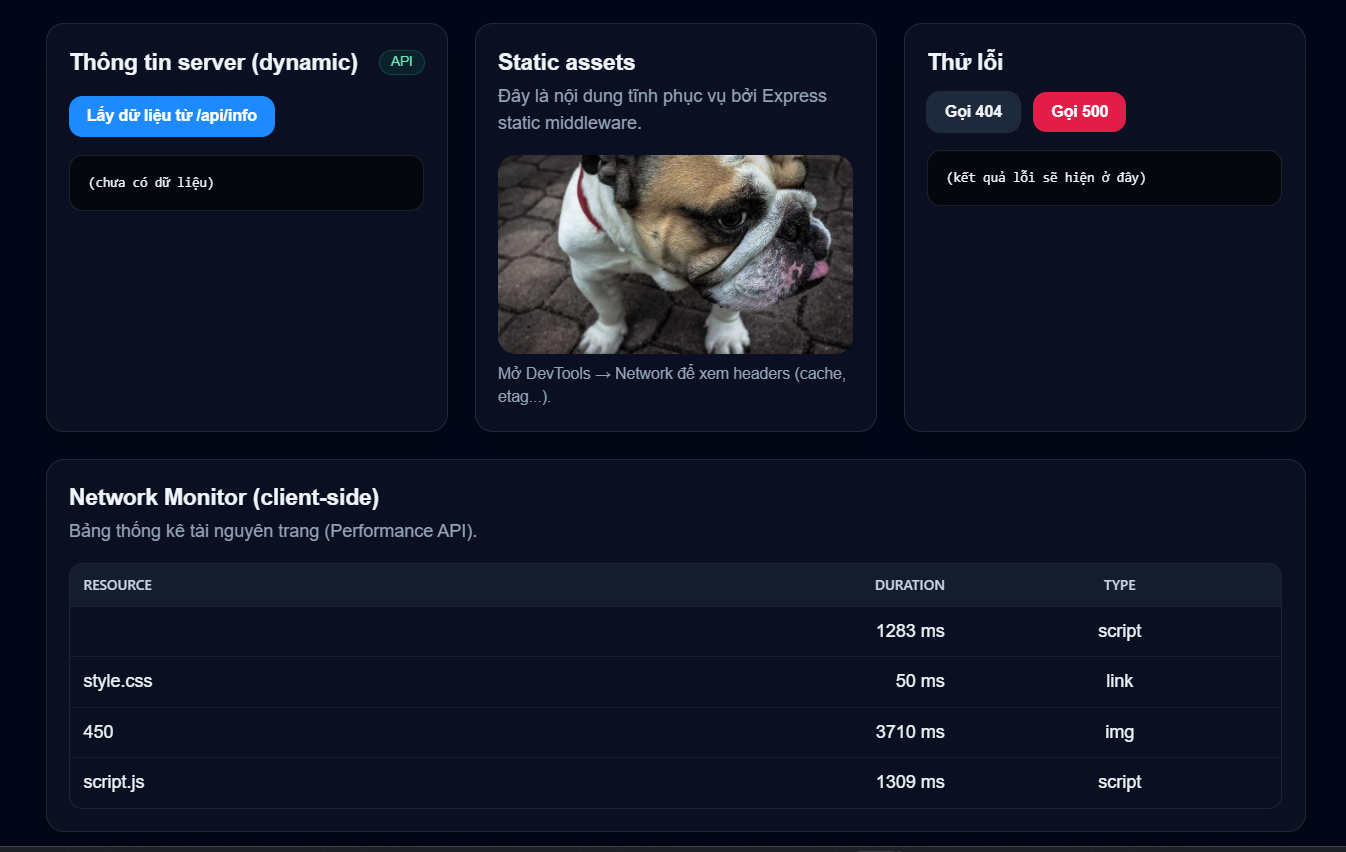
- `docs/technical-report.md`: Báo cáo kỹ thuật chi tiết.

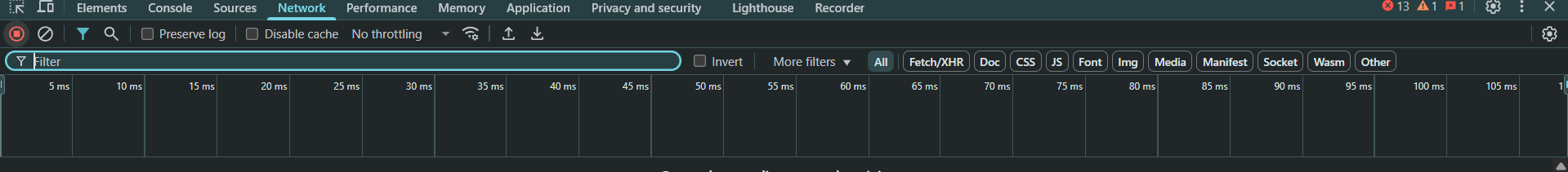
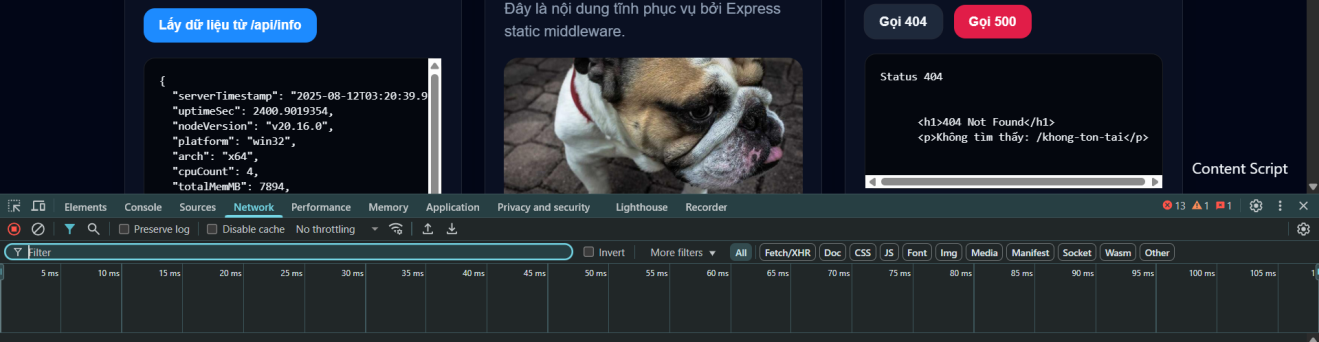
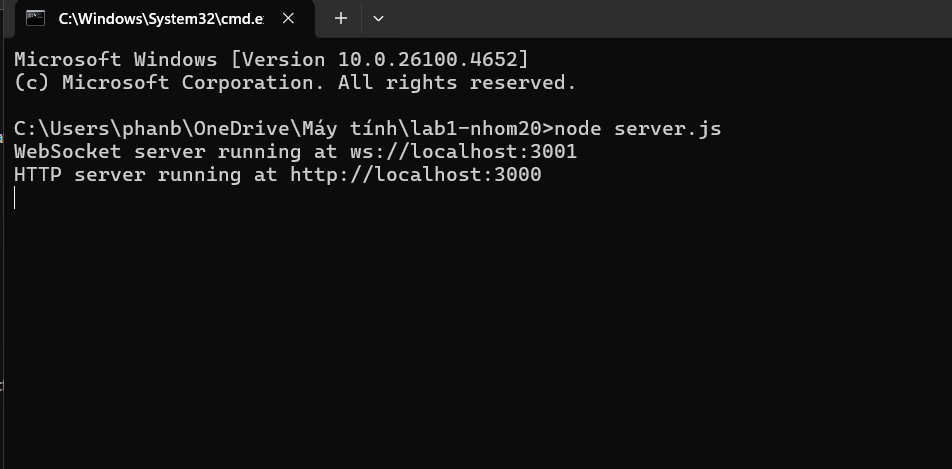
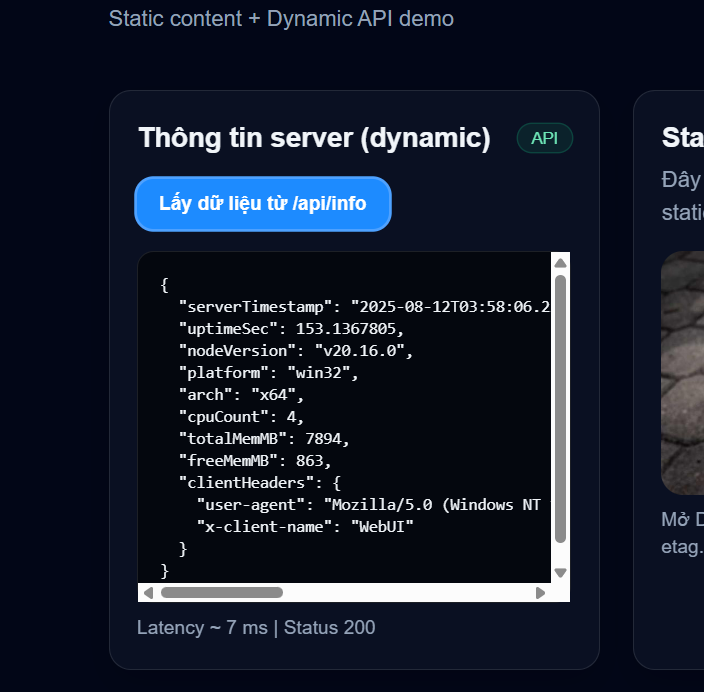
- `presentation/`: Slide trình bày dự án.

- `screenshots/`: Ảnh minh họa server chạy, network tab, kết quả API, phân tích hiệu suất.

---

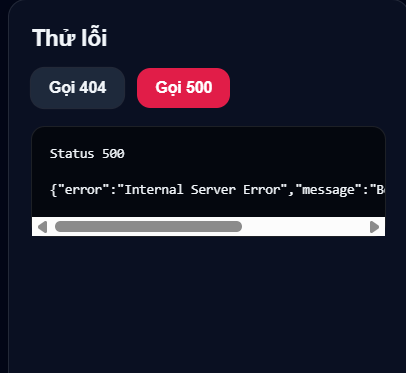
\*File này mô tả chi tiết toàn bộ code, chức năng từng file, cấu trúc dự án, giúp bạn nắm rõ khi thuyết trình hoặc báo cáo.\*

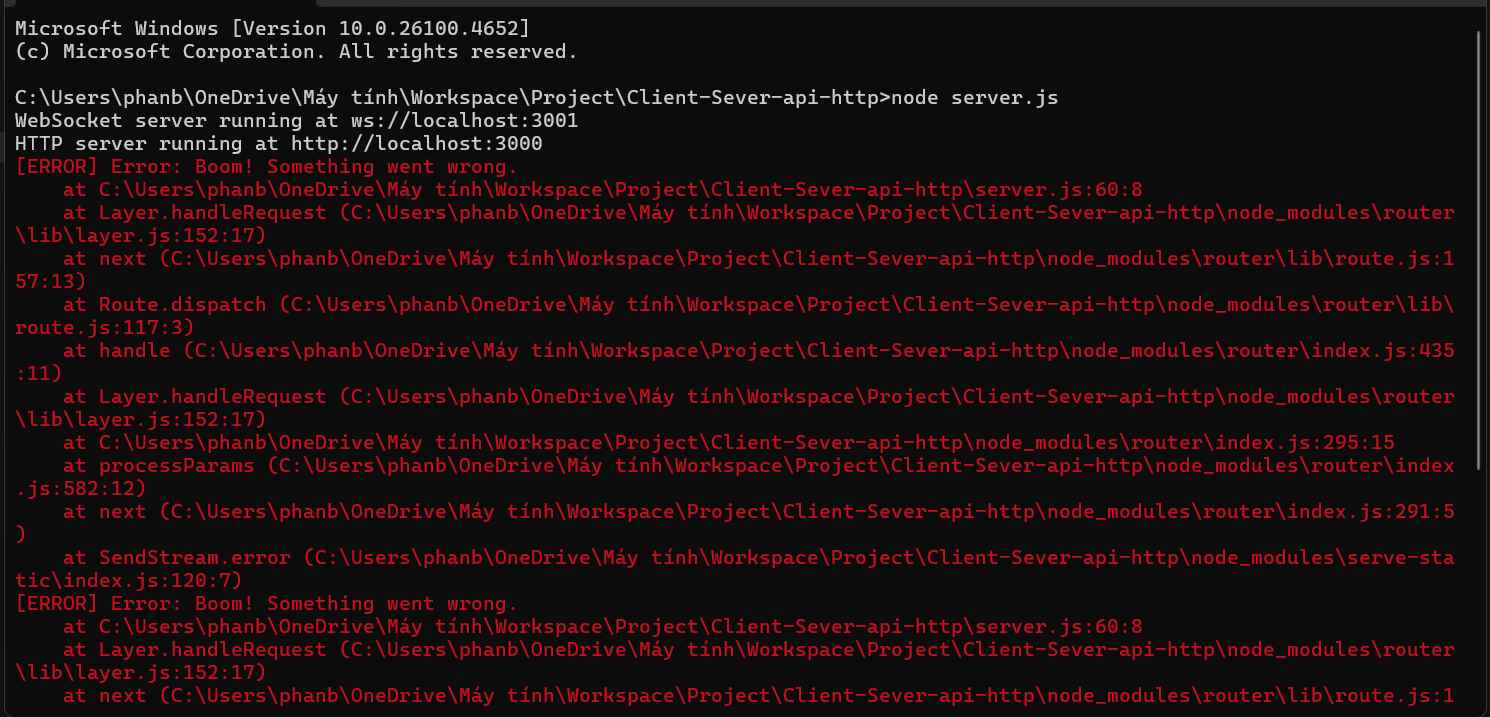




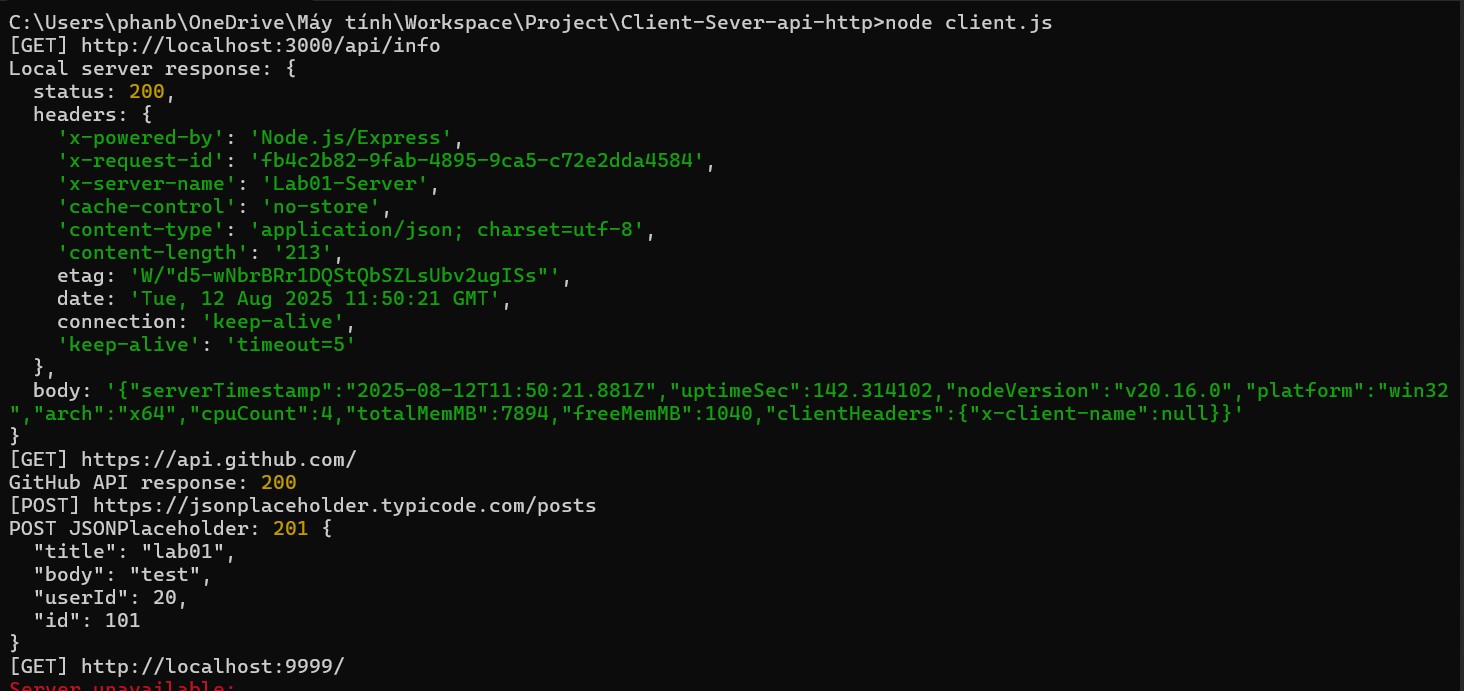
Khi gọi lỗi







Client.js



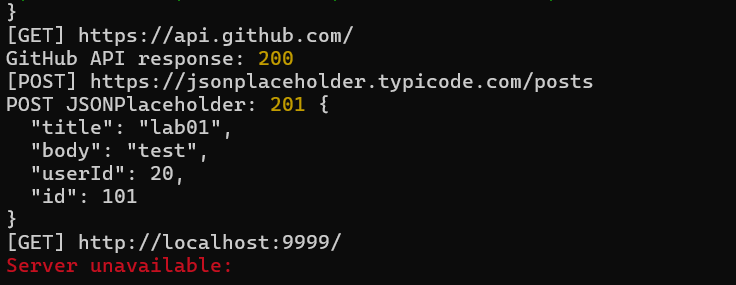
Trường hợp kiểm thử bắt buộc:

GET request tới server cục bộ

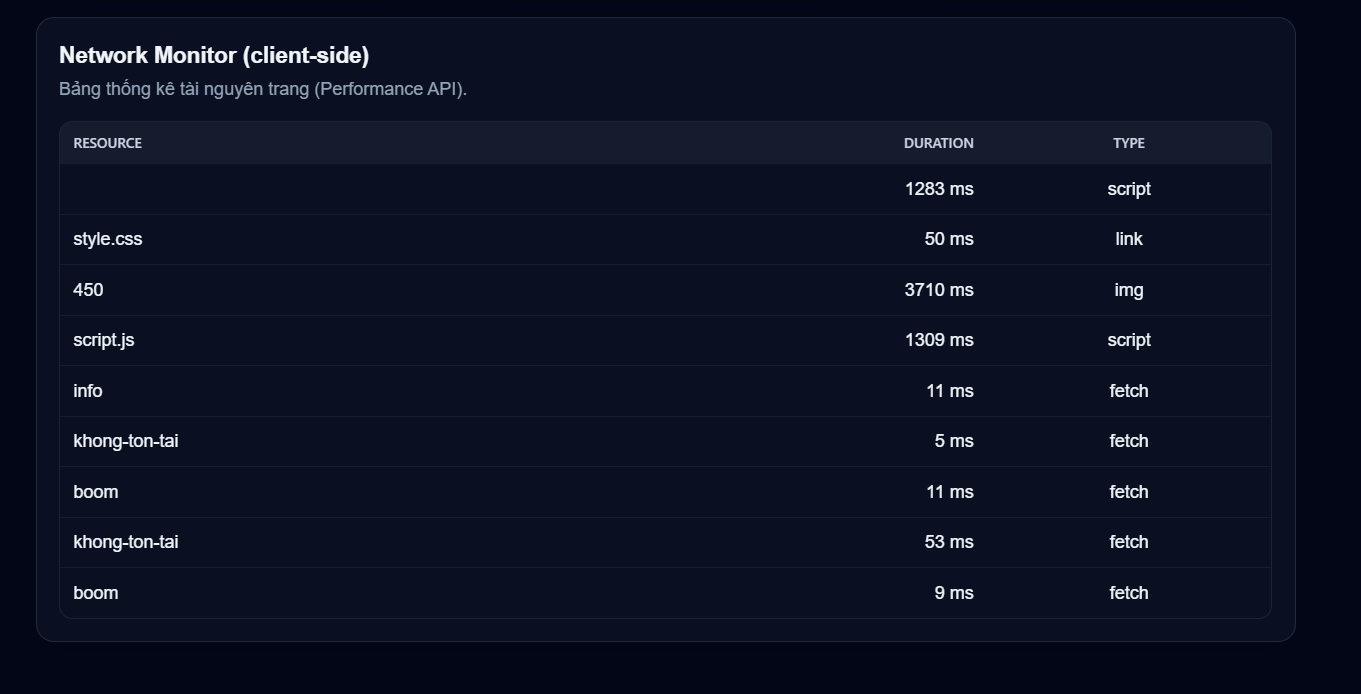
GET request tới external API (ví dụ: GitHub API)

POST request tới test endpoint (JSONPlaceholder)

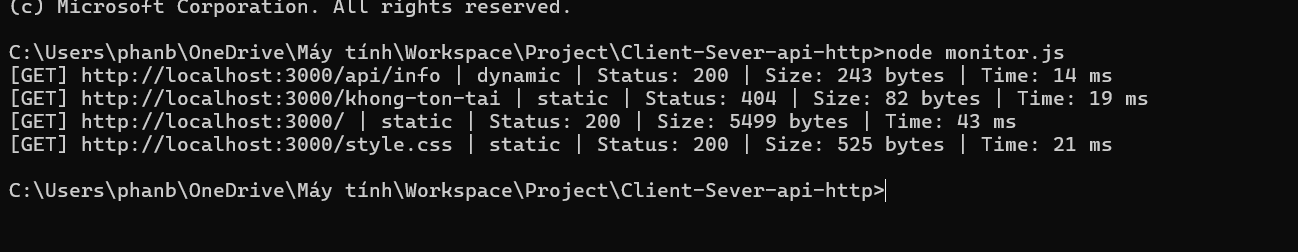
Xử lý lỗi khi server không khả dụng



Tiện ích giám sát mạng



Monitor.js



Nhiệm vụ thưởng

