Quiz 1 report
Technical Report

HATIM SALHI

February 16, 2026

Contents

1 Introduction

This report documents the analysis and resolution of several backend and frontend issues encountered during Sprint 1. The main objectives were:

- Identify backend errors and performance bottlenecks.
- Diagnose frontend XHR request failures.
- Optimize asset caching.
- Apply security measures against XSS using Content Security Policy (CSP).

2 Chapter 1 – Backend Challenges

2.1 Broken Route (/broken)

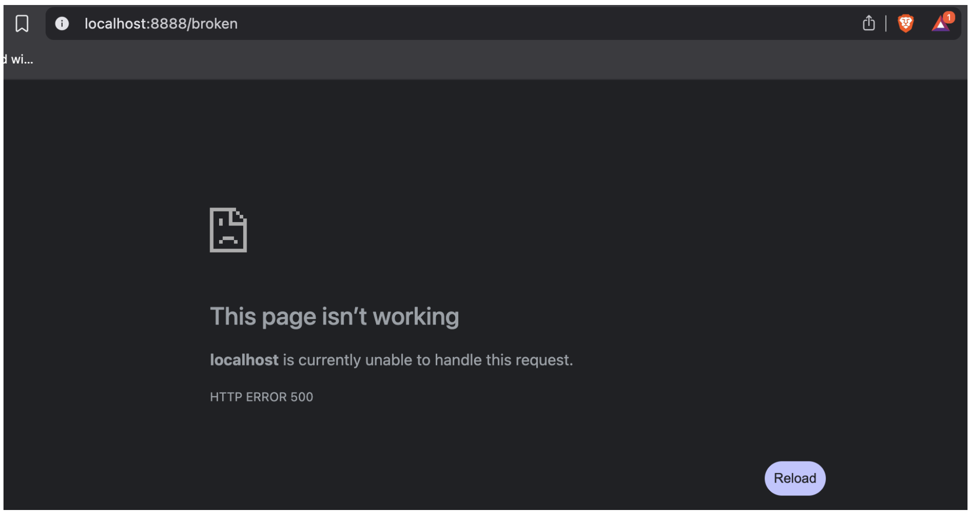
2.1.1 Problem Description

The route `/broken` returned a **500 Internal Server Error** with no visible error message.

2.1.2 Analysis

Error display was disabled in PHP configuration and a hidden Unicode character was present inside the method name `write()`, causing a fatal error.

2.1.3 Screenshot


localhost:8888/broken

This page isn't working

localhost is currently unable to handle this request.

HTTP ERROR 500

Figure 1: 500 Error on /broken route

2.1.4 Solution

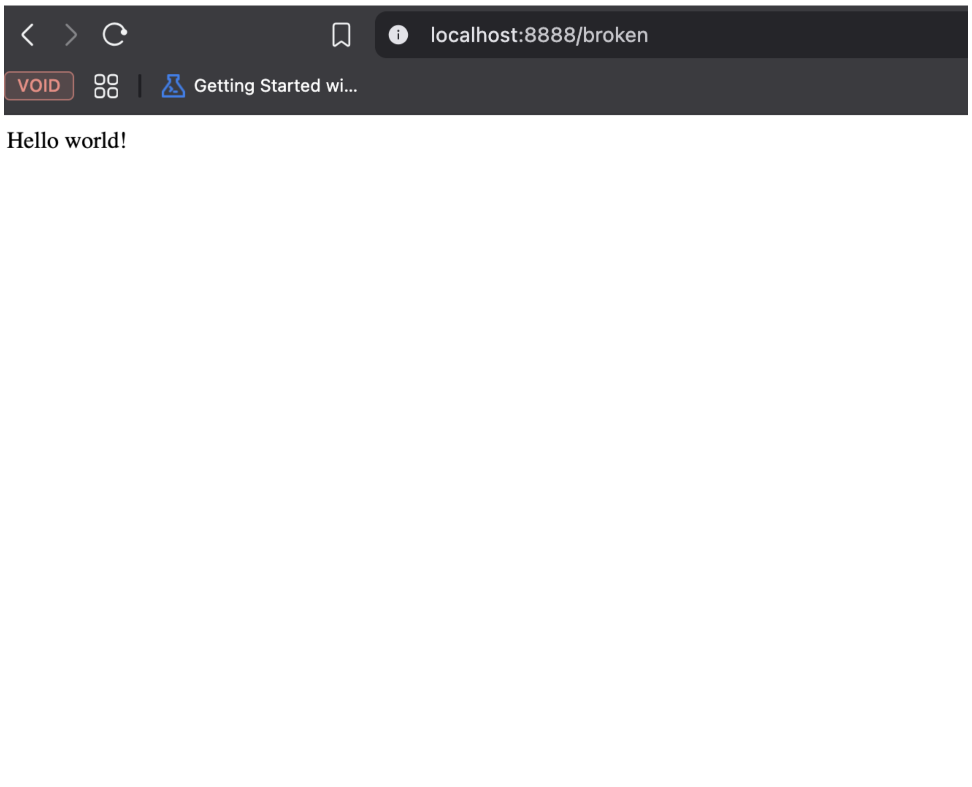
That write is not the normal write: there's a hidden Unicode character inside (write has a zero-width char). The line was rewritten correctly.

```
$response->getBody()->write("Hello␣world!");
```

2.1.5 Result

The route now returns a valid response.

2.1.6 Screenshot



Hello world!

Figure 2: Fixed Page

2.2 Crash Under Concurrent Requests (/crash)

2.2.1 Problem Description

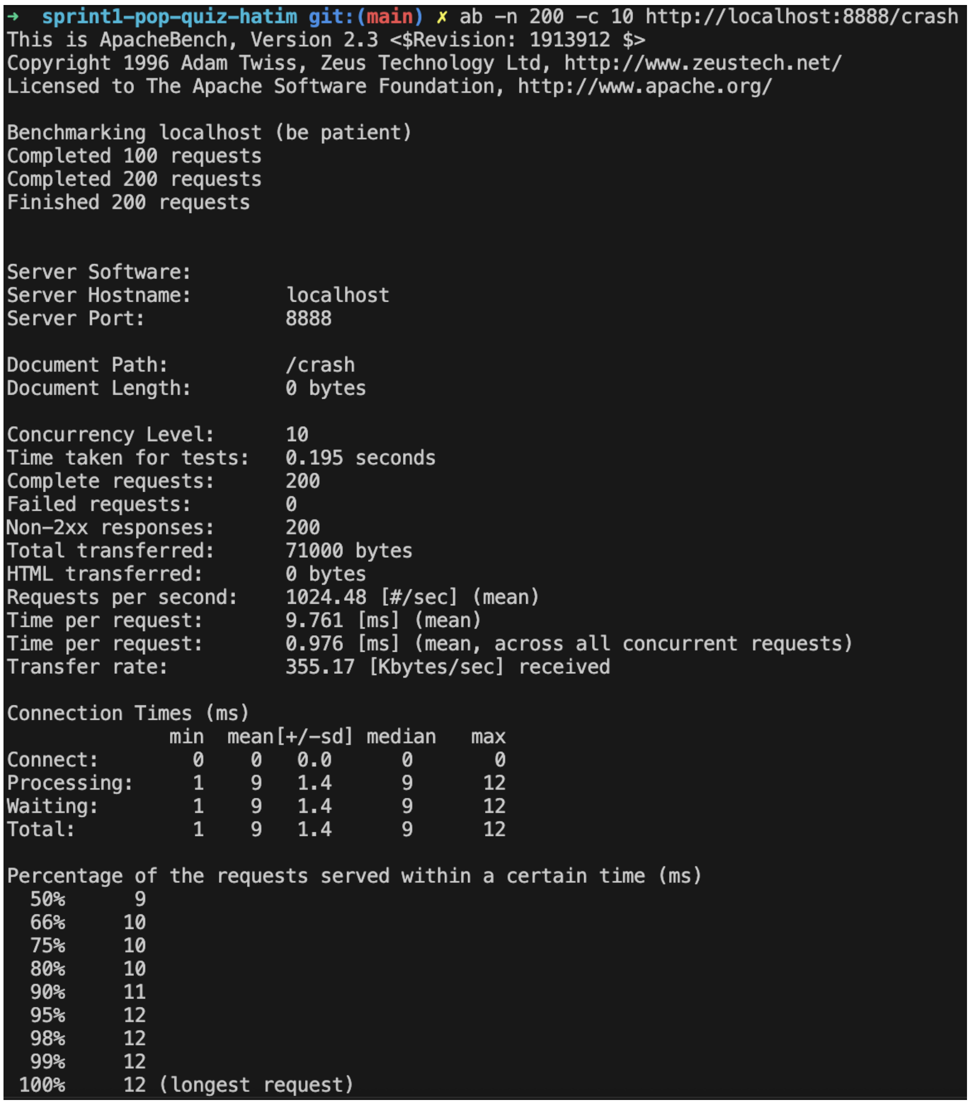
Using Apache Benchmark caused server crashes.

```
ab -n 200 -c 10 http://localhost:8888/crash
```

2.2.2 Analysis

The log rotation function read entire log files into memory and attempted to write large blocks of data, exceeding PHP's 8MB memory limit.

2.2.3 Screenshot



```
→  sprint1-pop-quiz-hatim git:(main) ✗ ab -n 200 -c 10 http://localhost:8888/crash
This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests


Server Software:
Server Hostname:        localhost
Server Port:            8888

Document Path:          /crash
Document Length:        0 bytes

Concurrency Level:      10
Time taken for tests:   0.195 seconds
Complete requests:      200
Failed requests:        0
Non-2xx responses:      200
Total transferred:      71000 bytes
HTML transferred:       0 bytes
Requests per second:    1024.48 [#/sec] (mean)
Time per request:       9.761 [ms] (mean)
Time per request:       0.976 [ms] (mean, across all concurrent requests)
Transfer rate:          355.17 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:     1    9   1.4      9      12
Waiting:        1    9   1.4      9      12
Total:          1    9   1.4      9      12

Percentage of the requests served within a certain time (ms)
  50%      9
  66%     10
  75%     10
  80%     10
  90%     11
  95%     12
  98%     12
  99%     12
 100%     12 (longest request)
```

Figure 3: Apache Benchmark command

2.2.4 Solution

Simplified rotation by truncating files.

```php
if (filesize($logFile) > 64 * 1024) {
    file_put_contents($logFile, "");
}
```

2.2.5 Result

Server remains stable under concurrent load.

3 Chapter 2 – Frontend Challenges

3.1 XHR Failure on /fetch

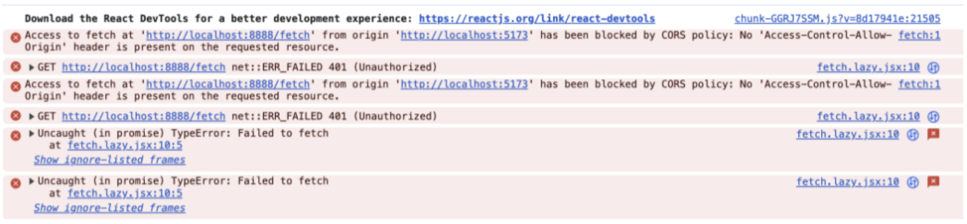
3.1.1 Problem Description

XHR requests returned 401 Unauthorized.

3.1.2 Analysis

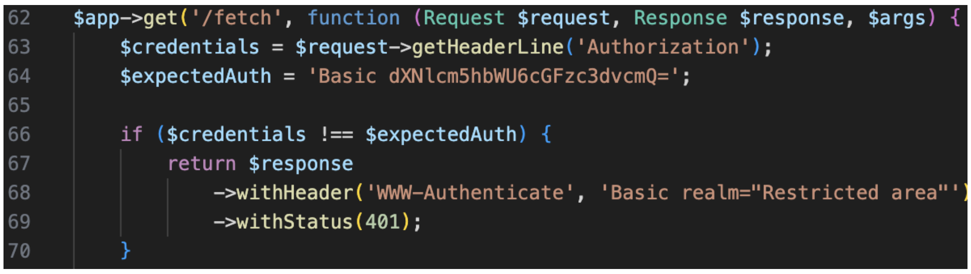
Backend requires a Basic Authorization header, but frontend did not send it.

3.1.3 Screenshot



```
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools    chunk-GGRJ7SSM.js?v=8d17941e:21505
Access to fetch at 'http://localhost:8888/fetch' from origin 'http://localhost:5173' has been blocked by CORS policy: No 'Access-Control-Allow-  fetch:1
Origin' header is present on the requested resource.
▶GET http://localhost:8888/fetch net::ERR_FAILED 401 (Unauthorized)                                            fetch.lazy.jsx:10
Access to fetch at 'http://localhost:8888/fetch' from origin 'http://localhost:5173' has been blocked by CORS policy: No 'Access-Control-Allow-  fetch:1
Origin' header is present on the requested resource.
▶GET http://localhost:8888/fetch net::ERR_FAILED 401 (Unauthorized)                                            fetch.lazy.jsx:10
▶Uncaught (in promise) TypeError: Failed to fetch                                                            fetch.lazy.jsx:10
    at fetch.lazy.jsx:10:5
  Show ignore-listed frames
▶Uncaught (in promise) TypeError: Failed to fetch                                                            fetch.lazy.jsx:10
    at fetch.lazy.jsx:10:5
  Show ignore-listed frames
```

Figure 4: 401 error on fetch request



```
$app->get('/fetch', function (Request $request, Response $response, $args) {
    $credentials = $request->getHeaderLine('Authorization');
    $expectedAuth = 'Basic dXNlcm5hbWU6cGFzc3dvcmQ=';

    if ($credentials !== $expectedAuth) {
        return $response
            ->withHeader('WWW-Authenticate', 'Basic realm="Restricted area"')
            ->withStatus(401);
    }
```

Figure 5: Backend Code

3.1.4 Solution

Add Authorization header in fetch.lazy.jsx (Frontend).

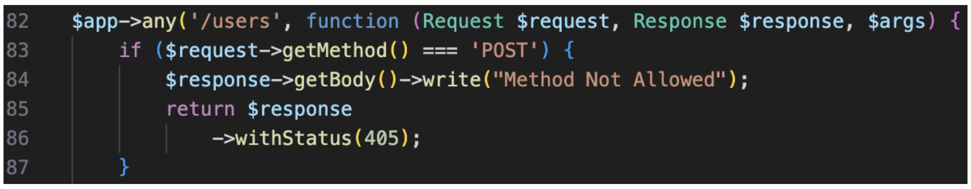
```
fetch(‘${API_URL}/fetch‘, {
  headers: {
    Authorization: ’Basic dXNlcm5hbWU6cGFzc3dvcmQ=’
  }
});
```

3.2 XHR Failure on /users

3.2.1 Problem Description

Frontend used POST while backend forbids POST.

3.2.2 Screenshot



```php
$app->any('/users', function (Request $request, Response $response, $args) {
    if ($request->getMethod() === 'POST') {
        $response->getBody()->write("Method Not Allowed");
        return $response
            ->withStatus(405);
    }
```

Figure 6: 405 Method Not Allowed

3.2.3 Solution

Switch request method to GET.

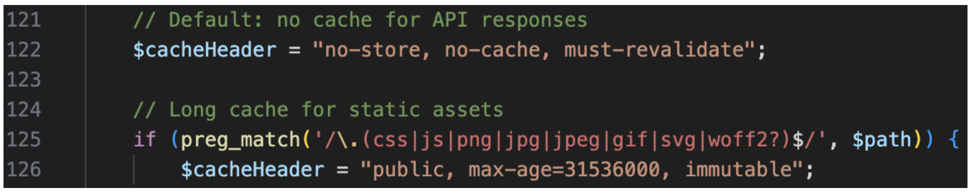
```
fetch(‘${API_URL}/users‘);
```

3.3 Asset Download Optimization

3.3.1 Problem Description

Assets were downloaded on every page refresh, which impacted performance.

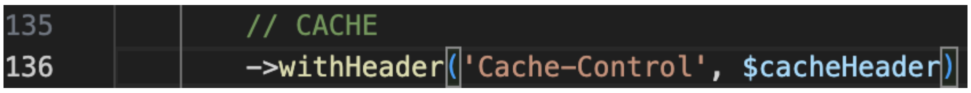
3.3.2 Screenshots and Solution



```php
    // Default: no cache for API responses
    $cacheHeader = "no-store, no-cache, must-revalidate";

    // Long cache for static assets
    if (preg_match('/\.(css|js|png|jpg|jpeg|gif|svg|woff2?)$/', $path)) {
        $cacheHeader = "public, max-age=31536000, immutable";
```

Figure 7: Solution



```php
        // CACHE
        ->withHeader('Cache-Control', $cacheHeader)
```

Figure 8: Solution (suite)

3.3.3 Result

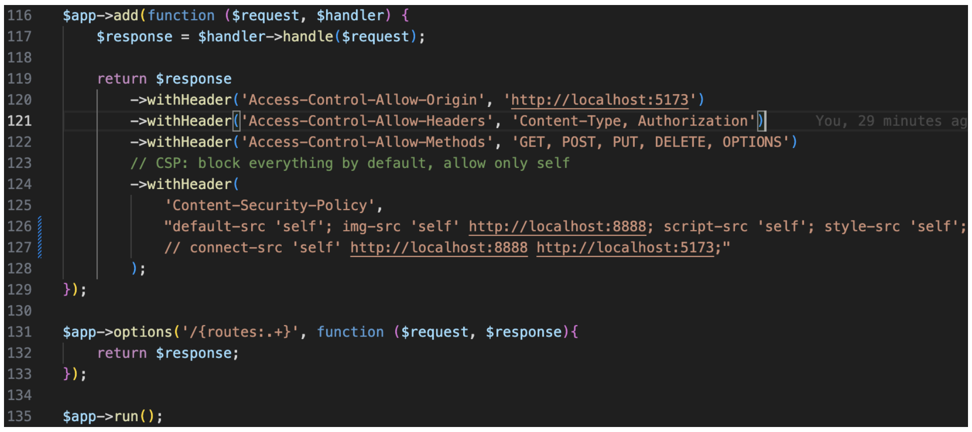
Browser caches assets efficiently.

3.4 XSS Prevention on /security

3.4.1 Problem Description

Page loads image from external untrusted source.

3.4.2 Screenshot



```php
$app->add(function ($request, $handler) {
    $response = $handler->handle($request);

    return $response
        ->withHeader('Access-Control-Allow-Origin', 'http://localhost:5173')
        ->withHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization')      You, 29 minutes ag
        ->withHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS')
        // CSP: block everything by default, allow only self
        ->withHeader(
            'Content-Security-Policy',
            "default-src 'self'; img-src 'self' http://localhost:8888; script-src 'self'; style-src 'self';
            // connect-src 'self' http://localhost:8888 http://localhost:5173;"
        );
});

$app->options('/{routes:.+}', function ($request, $response){
    return $response;
});

$app->run();
```

Figure 9: CSP blocking external image

3.4.3 Solution

Apply Content Security Policy.

```php
header("Content-Security-Policy: default-src 'self'; img-src 'self
    ';");
```

3.4.4 Result

Only images from localhost are allowed.

4 Conclusion

All identified issues were successfully diagnosed and resolved. The application now:

- Displays meaningful backend errors.
- Handles concurrent requests correctly.
- Allows authorized XHR calls.
- Uses optimized asset caching.
- Is protected against XSS via CSP.