Bug Analysis and Uptime Report

Objective

The purpose of this test was to analyze the `POST /api/name-checker` endpoint for its uptime and to identify any potential bugs related to the `name` parameter. The results were logged and analyzed over a continuous 10-minute monitoring session.

Testing Approach

1. **Endpoint Monitored:**
   - `POST /api/name-checker`
2. **Methodology:**
   - Requests were sent with various `name` values every second.
   - Responses were logged into a SQLite database (`request_logs_post.db`).
3. **Test Inputs:**
   - `"John"`: Valid name.
   - `"Alice"`: Another valid name.
   - `""`: Empty string.
   - `"1234"`: Numeric string.
   - `"!@#$%"`: Special characters.
   - `"中文测试"`: Non-ASCII characters.
4. **Duration:**
   - Monitoring was performed continuously for **10 minutes**.

Results

Total Requests: 432
Successful Requests (200): 390
Failed Requests (500): 42

Uptime Percentage:

(390 / 432) * 100 = 90.28%

Bug Observed

1. **Error Details:**

Occasionally, the service returned a `500 Internal Server Error` with the response:
{

"message": "System is down"

   - }
2. **Trigger Patterns:**
   - The `500` errors were sporadic but correlated with repeated or rapid requests.
   - The issue seemed to occur more frequently with edge-case inputs like:
     - Empty names: `""`
     - Non-ASCII characters: `"中文测试"`

Steps to Reproduce

1. Use the `monitor_post.rb` script to send requests with the test inputs.
   ruby scripts/monitor_post.rb
2. Monitor the database (`request_logs_post.db`) for entries with a `500` status code.
3. Alternatively, observe the console logs for:
   Logged POST response for name '<name>': 500

Recommendations

1. **Server-Side Improvements:**
   - Analyze server logs for underlying causes of the `500` errors.
   - Implement rate-limiting or caching to better handle high-frequency requests.
2. **Input Validation:**
   - Enhance input validation to handle edge cases like empty strings or special characters gracefully.
3. **Monitoring and Metrics:**
   - Continue monitoring to ensure uptime improvements after any fixes.

Conclusion

The service maintains a decent uptime of **90.28%** but struggles under prolonged or edge-case conditions. The sporadic `500` errors indicate a need for better load handling and validation mechanisms.