

LOAD TESTING

Using k6 by Grafana

Submitted By:

Nischal Shrestha

Submitted To:

Prajesh Shakya

Date: 17th April, 2025

ABSTRACT

This report details the methodology and findings of a comprehensive load testing conducted on “<https://groupiig.techarttrekkies.com.np/>” utilizing k6, a modern and developer-centric load testing tool by Grafana Labs. The primary objective was to evaluate the performance, stability, and scalability of the website under simulated real-world user traffic scenarios. This report outlines the design and execution of various test scripts, including simulating concurrent users, defining ramp-up stages, and implementing performance thresholds. Key Performance Indicators (KPIs) such as response time, request success rate, and error distribution were thoroughly measured and analyzed. The results provide valuable insights into the application’s behavior under different load levels, identifying potential bottlenecks and areas for optimization. Ultimately, this study demonstrates the effectiveness of k6 as a powerful tool for identifying performance limitations and ensuring the robustness of web applications under demanding user loads, contributing to improved user experience and system reliability.

TABLE OF CONTENTS

ABSTRACT.....	2
TABLE OF CONTENTS.....	3
k6 BY GRAFANA.....	4
Introduction.....	4
Installation.....	4
Basic Structure of a k6 Test.....	5
Testing.....	5
1. 1 Virtual user running for 5 minutes.....	5
2. 10 get HTTP requests to a Techart URL & wait for 1 second between requests.....	7
3. Load Testing (200 Virtual Users for 30m 30s).....	9
4. Stress Testing.....	12
a. Up to 1000 users.....	12
b. Up to 200 users.....	14
5. Spike Testing (300 Virtual Users for 7m).....	17
6. Soak Testing (50 Virtual Users for 40m).....	20
CONCLUSION.....	23

k6 BY GRAFANA

Introduction



Grafana k6 is an open-source, developer-friendly, and extensible load testing tool which allows us to prevent performance issues and proactively improve reliability. k6 helps developers simulate realistic user behavior and test how the systems behave as a result. Writing tests in k6 allows us to identify potential issues, such as slow response times or system failures, before they occur in production.

Using k6, we can test the reliability and performance of our application and infrastructure. It helps engineering teams prevent errors and SLO (Service Level Objective) breaches, enabling them to build resilient and high-performing applications that scale.

Engineering teams, including Developers, QA (Quality Assurance) Engineers, SDETs (Software Development Engineer in Test), and SREs (Site Reliability Engineering), commonly use k6 for:

- Load and Performance Testing
- Browser Performance Testing
- Performance and Synthetic Monitoring
- Automation of Performance Tests
- Chaos and Resilience Testing
- Infrastructure Testing

Installation

```
H:\>winget install k6.k6
Found k6 [k6.k6] Version 1.0.0-rc1
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Successfully verified installer hash
Starting package install...
Successfully installed
```

Basic Structure of a k6 Test

For k6 to be able to interpret and execute our test, every k6 script follows a common structure, revolving around a few core components:

1. Default function: This is where the test logic resides. It defines what our test will do and how it will behave during execution. It should be exported as the default function in our script.
2. Imports: We can import additional k6 modules or JavaScript libraries (jslibs) to extend our script's functionality, such as making HTTP requests or simulating browser interactions. *(P.S.: k6 is not built upon Node.js, and instead uses its own JavaScript runtime. Compatibility with some npm modules may vary.)*
3. Options (optional): Enable us to configure the execution of the test, such as defining the number of virtual users, the test duration, or setting performance thresholds.
4. Lifecycle operations (optional): Because our test might need to run code before and/or after the execution of the test logic, such as parsing data from a file, or download an object from Amazon S3, lifecycle operations allow us to write code, either as predefined functions or within specific code scopes, that will be executed at different stages of the test execution.

Testing

1. 1 Virtual user running for 5 minutes

Code:

```
import http from 'k6/http';
import { sleep } from 'k6';

export default function () {
  const res =
http.get('https://groupiig.techarttrekkies.com.np/');
  console.log('Status code: ${res.status}');
  sleep(1);
}
```

Output:

```
nischal.shrestha@acer-14 MINGW64 /c/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Intern/QA Testing (master)
$ k6 run script.js

      /\      Grafana
     /\  /\
    /\ /\  /\
   /\ /\ /\
  /\ /\ /\
 /\ /\ /\
/\ /\ /\

execution: local
script: script.js
output: -

scenarios: (100.00%) 1 scenario, 1 max VUs, 10m30s max duration (incl. graceful stop):
 * default: 1 iterations for each of 1 VUs (maxDuration: 10m0s, gracefulStop: 30s
)

█ TOTAL RESULTS

HTTP
http_req_duration.....: avg=26.38ms min=
26.38ms med=26.38ms max=26.38ms p(90)=26.38ms p(95)=26.38ms
{ expected_response:true }.....: avg=26.38ms min=
26.38ms med=26.38ms max=26.38ms p(90)=26.38ms p(95)=26.38ms
http_req_failed.....: 0.00% 0 out of
1
http_reqs.....: 1 0.943379/
s

EXECUTION
iteration_duration.....: avg=1.05s min=
1.05s med=1.05s max=1.05s p(90)=1.05s p(95)=1.05s
iterations.....: 1 0.943379/
s
vus.....: 1 min=1
max=1
vus_max.....: 1 min=1
max=1
max=1

NETWORK
data_received.....: 151 kB 143 kB/s
data_sent.....: 475 B 448 B/s

running (00m01.1s), 0/1 VUs, 1 complete and 0 interrupted iterations
default ✓ [=====] 1 VUs 00m01.1s/10m0s 1/1 iters, 1 per VU
```

Result:

- The duration of the HTTP requests:

Metrics	Duration	Expected Response
Average	26.38ms	26.38ms
Minimum	26.38ms	26.38ms
Median	26.38ms	26.38ms
Maximum	26.38ms	26.38ms
p(90)	26.38ms	26.38ms
p(95)	26.38ms	26.38ms

- 0.00% out of 1 HTTP request failed
- 1 HTTP request was made at a rate of 0.943379 req/second
- Iteration duration:
 - Average: 1.05s
 - Minimum: 1.05s
 - Median: 1.05s
 - Maximum: 1.05s
 - p(90): 1.05s
 - p(95): 1.05s
- Iterations: 1 iteration performed at a rate of 0.943379 iterations/second
- Virtual users: 1
- Maximum Virtual users: 1
- Data received: 151 kB, Average rate: 143 kB/s
- Data sent: 475 B, Average rate: 448 B/s
- Running: 1.1 s
- 0 out of 1 virtual user was active
- 1 iteration was successfully completed and 0 were interrupted

2. 10 get HTTP requests to a Techart URL & wait for 1 second between requests

Code:

```
//Import http module to make HTTP requests
import http from 'k6/http';
//Import sleep function to introduce delays
import { sleep } from 'k6';
export const options = {
  //Define the number of iterations for the test
  iterations: 10,
}
// The default exported function is gonna be picked up by k6 as the
entry point for the test script. It will be executed repeatedly in
"iterations" for the whole duration of the test.
```

```
export default function() {
  //make a get request to the target URL
  http.get('https://groupiig.techarttrekkies.com.np/');
  //Sleep for 1 second to simulate real-world usage
  sleep(1);
}
```

Output:

```
nischal.shrestha@acer-14 MINGW64 /c/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Intern/QA Testing (master)
$ k6 run load_testing.js.

      /\      Grafana
     /\  /\
    /\  /\  /\
   /\  /\  /\  /\
  /\  /\  /\  /\  /\

  execution: local
    script: load_testing.js.
    output: -

  scenarios: (100.00%) 1 scenario, 10 max VUs, 1m0s max duration (incl. graceful stop):
    * default: 10 looping VUs for 30s (gracefulStop: 30s)

  █ TOTAL RESULTS

  HTTP
    http_req_duration.....: avg=27.34ms min=12.91ms med=21.26ms max=219.43ms p(90)=28.43ms p(95)=40.33ms
    { expected_response:true }.....: avg=27.34ms min=12.91ms med=21.26ms max=219.43ms p(90)=28.43ms p(95)=40.33ms
    http_req_failed.....: 0.00% 0 out of 298
    http_reqs.....: 298 9.586154/s

  EXECUTION
    iteration_duration.....: avg=1.03s min=1.01s med=1.02s max=2.25s p(90)=1.03s p(95)=1.04s
    iterations.....: 298 9.586154/s
    vus.....: 3 min=3 max=10
    vus_max.....: 10 min=10 max=10

  NETWORK
    data_received.....: 44 MB 1.4 MB/s
    data_sent.....: 40 kB 1.3 kB/s

  running (0m31.1s), 00/10 VUs, 298 complete and 0 interrupted iterations
  default ✓ [=====] 10 VUs 30s
```

Result:

- The duration of the HTTP requests:

Metrics	Duration	Expected Response
Average	27.34ms	27.34ms

Minimum	12.91ms	12.91ms
Median	21.26ms	21.26ms
Maximum	219.43ms	219.43ms
p(90)	28.43ms	28.43ms
p(95)	40.33ms	40.33ms

- 0.00% out of 2 HTTP request failed
- 298 HTTP request was made at a rate of 9.586154 req/second
- Iteration duration:
 - Average: 1.03s
 - Minimum: 1.01s
 - Median: 1.02s
 - Maximum: 1.25s
 - p(90): 1.03s
 - p(95): 1.04s
- Iterations: 298 iteration performed at a rate of 9.586154 iterations/second
- Virtual users:
 - Minimum: 3
 - Maximum: 10
- Data received: 44 MB, Average rate: 1.4 MB/s
- Data sent: 40 kB, Average rate: 1.3 kB/s
- Running: 31.1s
- 0 out of 10 virtual user was active
- 298 iterations were successfully completed and 0 were interrupted

3. Load Testing (200 Virtual Users for 30m 30s)

Code:

```
import http from 'k6/http';
import { sleep } from 'k6';
```

```
export const options = {
  stages: [
    { duration: '5m', target: 200 }, //ramp up
    { duration: '20m', target: 200 }, //stable
    { duration: '5m', target: 0 }, //ramp down to 0 user
  ]
};

export default function () {
  http.get('https://groupiig.techarttrekkies.com.np/');
  sleep(1);
}
```

Output:

```
nischal.shrestha@acer-14 MINGW64 /c/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Intern/QA Testing (master)
$ k6 run load_testing.js

      /\      Grafana  /\
     /\  /\    /\  /\
    /\  /\  /\ /\  /\
   /\  /\  /\ /\  /\
  /\  /\  /\ /\  /\
 /\  /\  /\ /\  /\
/\  /\  /\ /\  /\

execution: local
script: load_testing.js
output: -

scenarios: (100.00%) 1 scenario, 200 max VUs, 30m30s max duration (incl. graceful stop):
  * default: Up to 200 looping VUs for 30m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)
WARN[0537] Request Failed      error="Get \"https://groupiig.techarttrekkies.com.np/\": request timeout"
WARN[0537] Request Failed      error="Get \"https://groupiig.techarttrekkies.com.np/\": request timeout"
WARN[0537] Request Failed      error="Get \"https://groupiig.techarttrekkies.com.np/\": request timeout"
WARN[0540] Request Failed      error="Get \"https://groupiig.techarttrekkies.com.np/\": request timeout"
WARN[1540] Request Failed      error="Get \"https://groupiig.techarttrekkies.com.np/\": request timeout"
WARN[1545] Request Failed      error="Get \"https://groupiig.techarttrekkies.com.np/\": request timeout"
WARN[1546] Request Failed      error="Get \"https://groupiig.techarttrekkies.com.np/\": request timeout"
WARN[1555] Request Failed      error="Get \"https://groupiig.techarttrekkies.com.np/\": request timeout"

TOTAL RESULTS
HTTP
  http_req_duration.....: avg=540.31ms min=0s med=401.92ms max=5.3s p(90)=1.21s p(95)=1.51s
  { expected_response:true }.....: avg=542.13ms min=18.44ms med=403.73ms max=5.3s p(90)=1.22s p(95)=1.51s
  http_req_failed.....: 0.34% 505 out of 146876
  http_reqs.....: 146876 81.572803/s

EXECUTION
  iteration_duration.....: avg=2.04s min=1.01s med=1.41s max=1m1s p(90)=2.28s p(95)=2.63s
  iterations.....: 146875 81.572247/s
  vus.....: 1 min=1 max=200
  vus_max.....: 200 min=200 max=200

NETWORK
  data_received.....: 22 GB 12 MB/s
  data_sent.....: 19 MB 10 kB/s

running (30m00.6s), 000/200 VUs, 146875 complete and 4 interrupted iterations
default ✓ [=====] 000/200 VUs 30m0s
```

Result:

- The duration of the HTTP requests:

Metrics	Duration	Expected Response
Average	540.31ms	542.13ms
Minimum	3s	16.44ms
Median	501.92ms	403.73ms
Maximum	5.3s	5.3s
p(90)	1.51s	1.51s
p(95)	1.51s	1.51s

- 0.34% totalling 505 failed out of 146,878 HTTP request failed
- 146,878 HTTP requests were made at an average rate of 81.57 req/second
- Iteration duration:
 - Average: 2.04s
 - Minimum: 1.01s
 - Median: 1.41s
 - Maximum: 1.5s
 - p(90): 2.63s
 - p(95): 2.63s
- Iterations: 146,875 iteration performed at a rate of 81.57 iterations/second
- Virtual users:
 - Minimum: 1
 - Maximum: 200
- Data received: 22 GB, Average rate: 12 MB/s
- Data sent: 19 MB, Average rate: 10 kB/s
- Running: 30m 0s
- All 200 virtual users had finished their execution
- 146,875 iterations were successfully completed and 4 were interrupted

4. Stress Testing

a. Up to 1000 users

Code:

```
import http from 'k6/http';
import { sleep } from 'k6';
export const options = {
  stages: [
    { duration: '1m', target: 200 }, //ramp up
    { duration: '5m', target: 200 }, //stable
    { duration: '1m', target: 800 }, //ramp up
    { duration: '5m', target: 200 }, //stable
    { duration: '1m', target: 1000 }, //ramp up
    { duration: '5m', target: 1000 }, //stable
    { duration: '5m', target: 0 }, //ramp down to 0 users
  ]
};
export default function () {
  http.get('https://groupiig.techarttrekkies.com.np/');
  sleep(1);
}
```

Output:

[illegible]

Result:

- The duration of the HTTP requests:

Metrics	Duration	Expected Response
Average	1.06s	1.21s
Minimum	0s	11ms
Median	68.53ms	0s
Maximum	7.66s	7.66s

p(90)	2.7s	2.7ms
p(95)	2.7s	2.7ms

- 99.99% totaling 50,250 out of 50,256 HTTP request failed
- 50,256 HTTP request was made at a rate of 67.8 req/second
- Iteration duration:
 - Average: 1.01s
 - Minimum: 11ms
 - Median: 24s
 - Maximum: 1m 1s
 - p(90): 2.2s
 - p(95): 2.2s
- Iterations: 50,250 iteration performed at a rate of 67.8 iterations/second
- Virtual users:
 - Minimum: 1
 - Maximum: 1000
- Data received: 6.5 GB, Average rate: 8.7 MB/s
- Data sent: 6.8 MB, Average rate: 9.1 kB/s
- Running: 12m 23.1s
- All 1000 virtual users had finished
- 50,250 iterations were completed (though likely with failures) and 536 were interrupted

b. Up to 200 users

Code:

```
import http from 'k6/http';
import { sleep } from 'k6';
export const options = {
  stages: [
    { duration: '1m', target: 50},
    { duration: '5m', target: 50},
```

```

    { duration: '1m', target: 100},
    { duration: '5m', target: 100},
    { duration: '1m', target: 150},
    { duration: '5m', target: 150},
    { duration: '1m', target: 200},
    { duration: '5m', target: 200},
  ]
};

export default function () {
  http.get('https://groupiig.techarttrekkies.com.np/');
  sleep (1);
}

```

Output:

```

nischal.shrestha@acer-14 MINGW64 /c/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Intern/QA Testing (master)
$ k6 run stress_testing200.js

      /\_/\
     /__  \
    (  oo  )
   /  _  \
  /  _  \
 /  _  \
/  _  \

Grafana

execution: local
script: stress_testing200.js
output: -

scenarios: (100.00%) 1 scenario, 200 max VUs, 24m30s max duration (incl. graceful stop):
* default: Up to 200 looping VUs for 24m0s over 8 stages (gracefulRampDown: 30s, gracefulStop: 30s)
WARN[1195] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1200] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1273] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1273] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1274] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1277] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1277] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1277] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1277] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1278] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"

```

```

MINGW64:/c:/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Int...
WARN[1313] Request Failed      error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1314] Request Failed      error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1315] Request Failed      error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1315] Request Failed      error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1315] Request Failed      error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[1321] Request Failed      error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"

TOTAL RESULTS

HTTP
http_req_duration.....: avg=325.08ms min
=0s    med=161.36ms max=9.11s p(90)=860.04ms p(95)=1.1s
{ expected_response:true }.....: avg=325.22ms min
=8.43ms med=161.67ms max=9.11s p(90)=860.11ms p(95)=1.1s
http_req_failed.....: 0.04% 50 out of
105953
http_reqs.....: 105953 79.860986
/s

EXECUTION
iteration_duration.....: avg=1.41s    min
=1s    med=1.16s    max=1m1s    p(90)=1.87s    p(95)=2.12s
iterations.....: 105875 79.802195
/s

vus.....: 200    min=1
max=200
vus_max.....: 200    min=200
max=200

NETWORK
data_received.....: 16 GB 12 MB/s
data_sent.....: 13 MB 10 kB/s

running (22m06.7s), 000/200 VUs, 105875 complete and 200 interrupted iterations
default x [=====>----] 130/200 VUs 22m06.7s/24m00.0s
ERRO[1327] test run was aborted because k6 received a 'interrupt' signal

```

Result:

- The duration of the HTTP requests:

Metrics	Duration	Expected Response
Average	325.08ms	325.22ms
Minimum	0s	8.43ms
Median	161.36ms	161.67ms
Maximum	9.11s	9.11s
p(90)	860.04ms	860.11ms
p(95)	1.1s	1.1s

- 0.04% totaling 50 out of 1,05,953 HTTP request failed
- 1,05,875 HTTP request was made at a rate of 79.860 req/second
- Iteration duration:
 - Average: 1.41s
 - Minimum: 1s
 - Median: 1.16s
 - Maximum: 1m 1s
 - p(90): 1.87s
 - p(95): 2.12s
- Iterations: 1,05,875 iteration performed at a rate of 79.80 iterations/second
- Virtual users:
 - Minimum: 1
 - Maximum: 200
- Data received: 16 GB, Average rate: 12 MB/s
- Data sent: 13 MB, Average rate: 10 kB/s
- Running: 22m 06.7s
- 130 virtual users had finished
- 1,05,875 iterations were completed and 200 were interrupted

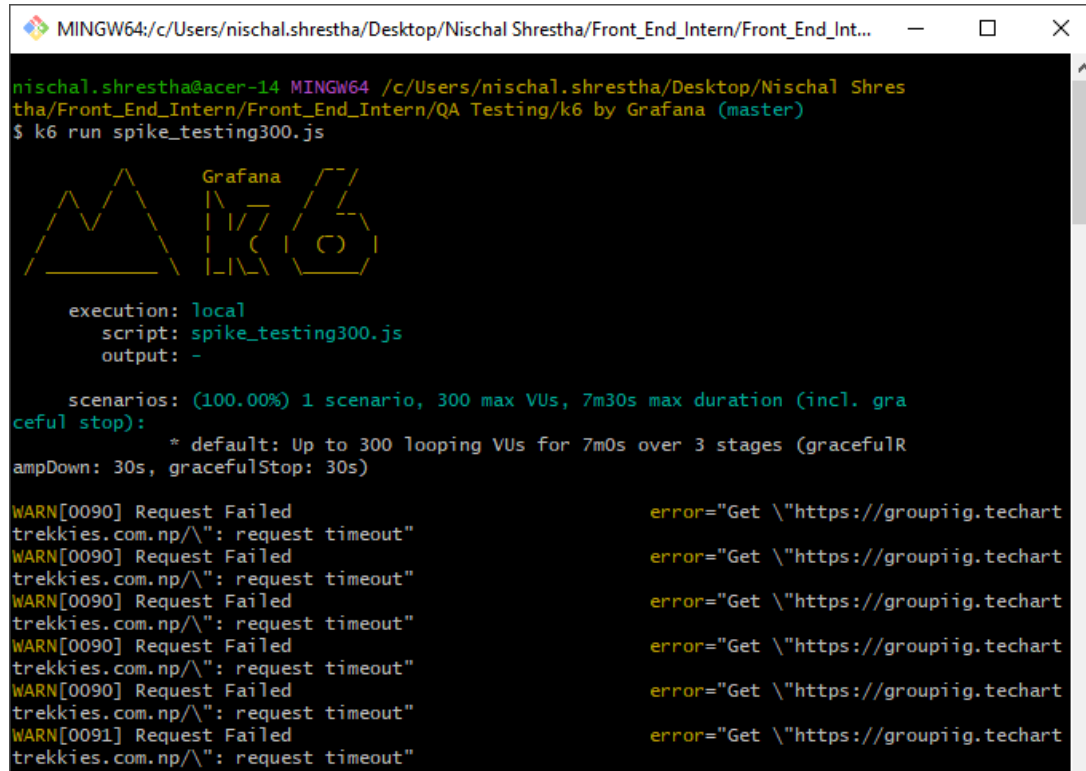
5. Spike Testing (300 Virtual Users for 7m)

Code:

```
import http from 'k6/http';
import { sleep } from 'k6';
export const options = {
  stages: [
    { duration: '1m', target: 300},
    { duration: '5m', target: 300},
    { duration: '1m', target: 0},
  ]
};
export default function () {
  http.get('https://groupiig.techarttrekkies.com.np/');
```

```
sleep (1);  
}
```

Output:



```
MINGW64:/c:/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Int...  
nischal.shrestha@acer-14 MINGW64 /c:/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Intern/QA Testing/k6 by Grafana (master)  
$ k6 run spike_testing300.js  
  
      execution: local  
      script: spike_testing300.js  
      output: -  
  
      scenarios: (100.00%) 1 scenario, 300 max VUs, 7m30s max duration (incl. graceful stop):  
        * default: Up to 300 looping VUs for 7m0s over 3 stages (graceful RampDown: 30s, gracefulStop: 30s)  
  
WARN[0090] Request Failed      error="Get \"https://groupiig.techart  
trekkies.com.np/\": request timeout"  
WARN[0090] Request Failed      error="Get \"https://groupiig.techart  
trekkies.com.np/\": request timeout"  
WARN[0090] Request Failed      error="Get \"https://groupiig.techart  
trekkies.com.np/\": request timeout"  
WARN[0090] Request Failed      error="Get \"https://groupiig.techart  
trekkies.com.np/\": request timeout"  
WARN[0090] Request Failed      error="Get \"https://groupiig.techart  
trekkies.com.np/\": request timeout"  
WARN[0091] Request Failed      error="Get \"https://groupiig.techart  
trekkies.com.np/\": request timeout"
```

```

WARN[0119] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"
WARN[0120] Request Failed                                error="Get \"https://groupiig.techart
trekkies.com.np/\": request timeout"

■ TOTAL RESULTS

HTTP
http_req_duration.....: avg=578.15ms min
=0s    med=129.1ms max=34.83s p(90)=466.47ms p(95)=611.97ms
{ expected_response:true }.....: avg=587.01ms min
=19.79ms med=131ms max=34.83s p(90)=467.75ms p(95)=620.95ms
http_req_failed.....: 1.50% 151 out o
f 10010
http_reqs.....: 10010 83.298281
/s

EXECUTION
iteration_duration.....: avg=2.47s min
=1.02s med=1.13s max=1m1s p(90)=1.47s p(95)=1.87s
iterations.....: 9899 82.374594
/s
vus.....: 300 min=5
max=300
vus_max.....: 300 min=300
max=300

NETWORK
data_received.....: 1.5 GB 12 MB/s
data_sent.....: 1.3 MB 11 kB/s

running (2m00.2s), 000/300 VUs, 9899 complete and 300 interrupted iterations
default x [=====>-----] 288/300 VUs 2m00.2s/7m00.0s
ERRO[0120] test run was aborted because k6 received a 'interrupt' signal

```

Result:

- The duration of the HTTP requests:

Metrics	Duration	Expected Response
Average	578.15ms	587.01ms
Minimum	0s	19.79ms
Median	129.1ms	131ms
Maximum	34.83s	34.83s
p(90)	466.47ms	467.75ms
p(95)	611.97ms	620.95ms

- 1.50% totalling 151 failed out of 10,010 HTTP request failed
- 10,010 HTTP requests were made at an average rate of 83.2982 req/second
- Iteration duration:

- Average: 2.47s
- Minimum: 1.02s
- Median: 1.13s
- Maximum: 1m 1s
- p(90): 1.47s
- p(95): 1.87s
- Iterations: 9,899 iteration performed at a rate of 82.374 iterations/second
- Virtual users:
 - Minimum: 5
 - Maximum: 300
- Data received: 1.5 GB, Average rate: 12 MB/s
- Data sent: 1.3 MB, Average rate: 11 kB/s
- Running: 2m 0.2s
- 288 virtual users had finished their execution
- 9,899 iterations were completed and 300 were interrupted

6. Soak Testing (50 Virtual Users for 40m)

Code:

```
import http from 'k6/http';
import { sleep } from 'k6';
export const options = {
  stages: [
    { duration: '5m', target: 50},
    { duration: '30m', target: 50},
    { duration: '5m', target: 0},
  ]
};
export default function () {
  http.get('https://groupiig.techarttrekkies.com.np/');
  sleep (1);
}
```

Output:

```
MINGW64/c:/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Intern/QA Testing/k6 by Grafana
nischal.shrestha@acer-14 MINGW64 /c:/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Intern/QA Testing/k6 by Grafana (master)
$ k6 run soak_testing50vu.js

      Grafana
    /  /  /
   /  /  /
  /  /  /

execution: local
script: soak_testing50vu.js
output: -

scenarios: (100.00%) 1 scenario, 50 max VUs, 40m30s max duration (incl. graceful stop):
  * default: Up to 50 looping VUs for 40m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

TOTAL RESULTS
HTTP
http_req_duration.....: avg=45.81ms min=7.97ms med=22.8ms max=6.53s p(90)=58.96ms p(95)=115.31ms
  { expected_response:true }.....: avg=45.81ms min=7.97ms med=22.8ms max=6.53s p(90)=58.96ms p(95)=115.31ms
http_req_failed.....: 0.00% 0 out of 100491
http_reqs.....: 100491 41.863747/s

EXECUTION
iteration_duration.....: avg=1.04s min=1s med=1.02s max=7.53s p(90)=1.06s p(95)=1.11s
iterations.....: 100491 41.863747/s
vus.....: 1 min=1 max=50
vus_max.....: 50 min=50 max=50

NETWORK
data_received.....: 15 GB 6.2 MB/s
data_sent.....: 13 MB 5.3 kB/s

running (40m00.4s), 00/50 VUs, 100491 complete and 0 interrupted iterations
default ✓ [=====] 00/50 VUs 40m0s
nischal.shrestha@acer-14 MINGW64 /c:/Users/nischal.shrestha/Desktop/Nischal Shrestha/Front_End_Intern/Front_End_Intern/QA Testing/k6 by Grafana (master)
$ |
```

Result:

- The duration of the HTTP requests:

Metrics	Duration	Expected Response
Average	45.81ms	45.81ms
Minimum	7.97s	7.97ms
Median	22.8ms	22.8ms
Maximum	6.53s	6.53s
p(90)	58.96ms	58.96ms
p(95)	115.31ms	115.31ms

- 0.00% totalling 0 failed out of 1,00,491 HTTP request failed
- 1,00,491 HTTP requests were made at an average rate of 41.86 req/second
- Iteration duration:

- Average: 1.04s
 - Minimum: 1s
 - Median: 1.02s
 - Maximum: 7.53s
 - p(90): 1.06s
 - p(95): 1.11s
- Iterations: 1,00,491 iteration performed at a rate of 41.86 iterations/second
- Virtual users:
 - Minimum: 1
 - Maximum: 50
- Data received: 15 GB, Average rate: 6.2 MB/s
- Data sent: 13 MB, Average rate: 5.3 kB/s
- Running: 40m 0.4s
- 50 virtual users had finished their execution
- 1,00,491 iterations were completed and 0 were interrupted

CONCLUSION

The Load Testing results indicate that “<https://groupiig.techarttrekkies.com.np/>” demonstrates reasonable performance and stability under moderate and sustained loads (up to 200 virtual users). However, the application exhibits significant performance degradation and high failure rates when subjected to higher concurrent users (especially 1000) and sudden traffic spikes which need significant optimizations. Addressing the identified bottlenecks and implementing strategies to handle peak loads will be critical for ensuring a consistent and positive user experience.