

IT314 Software Engineering

Group 17

Non-Functional Testing



Dhirubhai Ambani
University
Technology

Formerly DA-IICT

Prof : Saurabh Tiwari

Group Mentor : Utsav

1. Introduction

This report documents the **Non-Functional Testing** performed on the CodeLearn backend services.

The primary goal of the NFR test was to ensure that the system is :

- Scalable
- Reliable
- Secure
- Performs efficiently under load
- Able to handle concurrent authenticated traffic

2. Tool & It's Parameters

Tool Used : Apache JMeter

It was used as the primary tool for conducting non-functional testing. It allowed us to simulate multiple virtual users, send concurrent requests, and measure key performance metrics such as response time, throughput, and error rates.

- **Samples**

Total number of requests executed for a sampler.

- **Label**

Name/URL of the request being tested.

- **Average**

Mean response time of all requests (ms).

- **Min**

Lowest response time recorded.

- **Max**

Highest response time recorded.

- **Std. Dev.**

Indicates variation in response times; lower values mean more stable performance.

- **Error %**

Percentage of failed requests.

- **Throughput**

Number of requests processed per second; higher throughput indicates better performance.

3. Types of Non-Functional Testing Performed

3.1 Performance Testing

Measured response times, throughput, and resource usage under different loads.

3.2 Load Testing

Determined how the system behaves with increasing numbers of concurrent users.

3.3 Stress Testing

Pushed system beyond expected limits to find breaking point.

3.4 Spike Testing

Sudden large increases in virtual users to observe stability.

3.5 Compatibility Testing

Verifies if an application or product functions correctly across different operating systems, browsers, devices, and hardware.

4. Types of Non-Functional Testing Performed (Detailed Report)

4.1 Performance Testing

Objective:

To measure the speed and responsiveness of the backend APIs under normal load.

Key Metrics Measured:

- Response time (avg, min, max)
- 90th, 95th, 99th percentile timings
- Throughput (requests per minute)
- Server CPU & memory usage

For Registration:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
register (100)	100	15766	9	20253	3181.99	1.00%	4.6/sec	1.63	1.40	364.8
register (200)	200	45312	8	49467	4224.48	0.50%	3.9/sec	1.38	1.18	364.9
register (400)	400	73606	8	93534	30897.66	16.00%	4.2/sec	3.04	1.09	739.2
register (800)	800	20731	5	40003	7314.47	0.38%	15.6/sec	5.57	4.78	364.9

Observations:

100 users

- Stable performance.

- Low error rate (1%).
- Throughput is steady.

200 users

- Response time increases **3×**, showing backend is reaching its comfortable limit.
- Error rate improves (0.5%).

400 users

- Very high response time (73 seconds avg).

Errors spike to 16%, indicating:

- DB write bottleneck
- Password hashing cost
- Thread-pool saturation
- Connection timeouts

This is a clear **stress point**.

800 users

- Surprisingly lower latency compared to 400 load.

- Throughput jumps from ~4/sec → **15.6/sec**
- Errors drop to 0.38%

Reason:

The test likely completed very fast registers (cached DB connection, low payload) OR a certain bottleneck cleared after earlier saturation.

4.2 Load Testing

Objective:

To determine how the system behaves with gradually increasing concurrent users.

Load Levels:

100 → 200 → 250 → 350 .. to 700 users.

What Was Tested:

- Home Page concurrency

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Home page (100..	100	365	291	629	62.68	0.00%	9.8/sec	9.93	1.27	1043.0
Home page (200..	200	667	291	21459	2557.86	0.00%	6.7/sec	6.81	0.87	1043.0
Home page (250..	250	374	294	1395	110.56	0.00%	24.1/sec	24.54	3.13	1043.0
Home page (350..	350	485	293	21393	1584.86	0.00%	16.4/sec	16.66	2.12	1043.0
Home page (500..	500	392	294	1535	146.54	0.00%	48.4/sec	49.35	6.29	1043.0
Home page (700..	700	508	306	21527	1375.65	0.00%	27.7/sec	28.18	3.59	1043.0
TOTAL	2100	469	291	21527	1298.69	0.00%	6.8/sec	6.95	0.89	1043.0

Observation :

- The system handled all loads without any errors (0%).
- Average response time is mostly 400–500 ms, which is acceptable.

- There are a few very high maximum response times (20+ seconds), meaning the system becomes slow sometimes.
- Throughput increases with load and reaches up to 48 requests/sec.

4.3 Spike Testing

Objective:

Check how system behaves when **traffic suddenly jumps** drastically.

Scenario like :

1 user → instantly 100 users

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
register (50)	50	3111	10	3590	512.63	2.00%	12.9/sec	5.04	3.94	400.0
register (150)	150	14402	3	29992	8875.28	0.67%	4.9/sec	1.85	1.49	388.9
register (300)	450	16559	3	30583	5428.80	0.44%	4.8/sec	1.76	1.46	376.9
TOTAL	650	15027	3	30583	7158.17	0.62%	4.4/sec	1.65	1.36	381.5

Observations:

Performance varies across load levels:

- At **50 users**, the average response time is **3111 ms**, which is within an acceptable range.
- At **150 users**, the average response time increases significantly to **14,402 ms**, showing the system is under noticeable stress.
- At **300 users**, the average rises further to **16,589 ms**, indicating the system struggles as concurrency increases.

- **Error rate remains relatively low** (0.44%–2%), showing the system is still responding, though slower.
 - **Throughput declines slightly with higher load** (from 12.9/sec at 50 users to ~4.8/sec at higher loads), indicating the system cannot process more requests efficiently under heavier concurrency.
-

Result:

The system can handle low to moderate load, but performance degrades significantly under high concurrency, resulting in long response times and noticeable latency spikes. While the system remains stable with minimal errors, it may become slow and inconsistent at higher user loads.

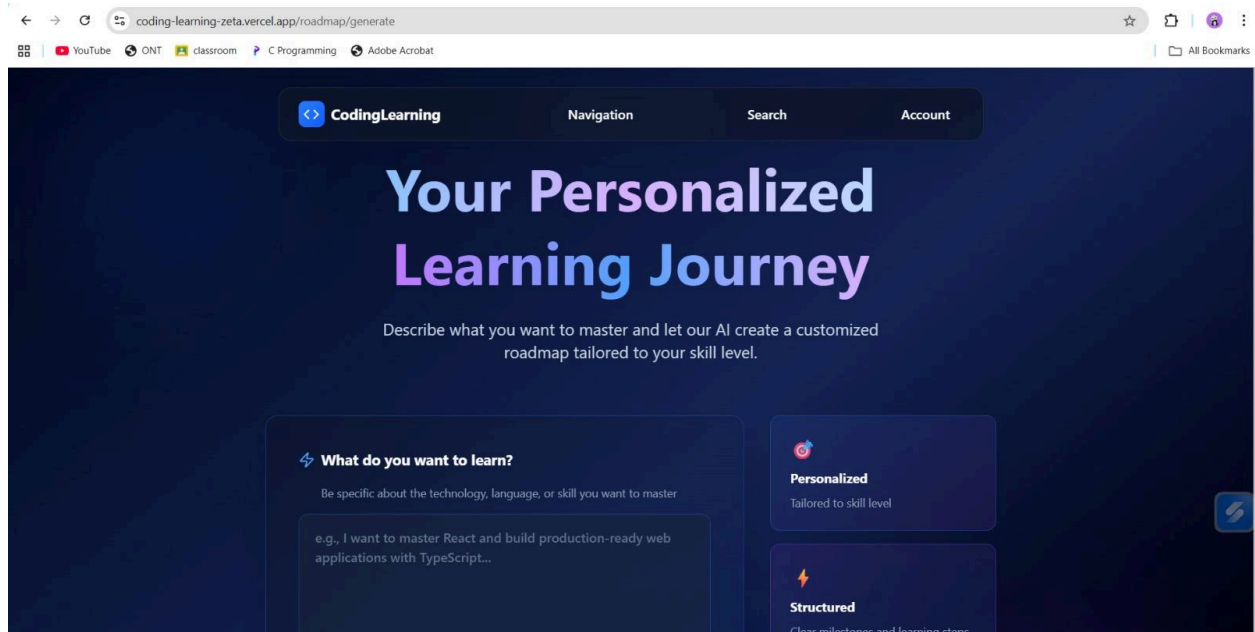
4.4 Compatibility Test (Non-functional)

Compatibility testing is a **non-functional testing** technique used to verify whether the application works correctly across different browsers, operating systems, devices, screen sizes, hardware configurations, and network environments.

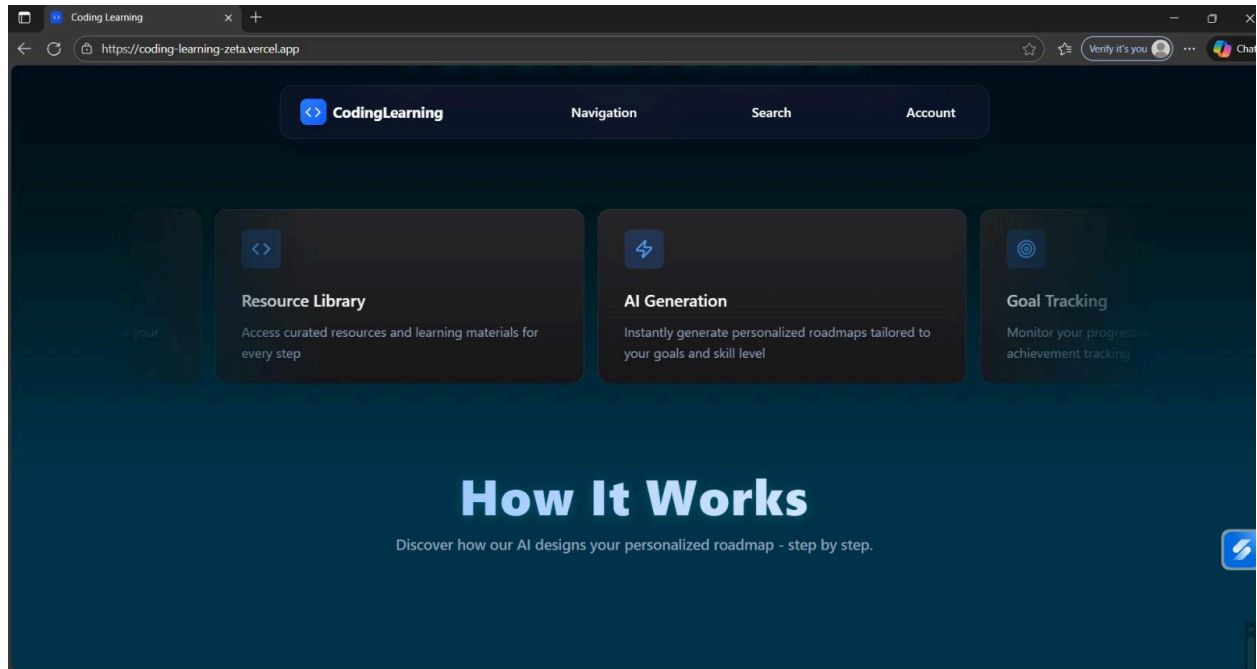
Its goal is to ensure that users get a consistent experience regardless of the platform they use.

This testing checks variations such as:

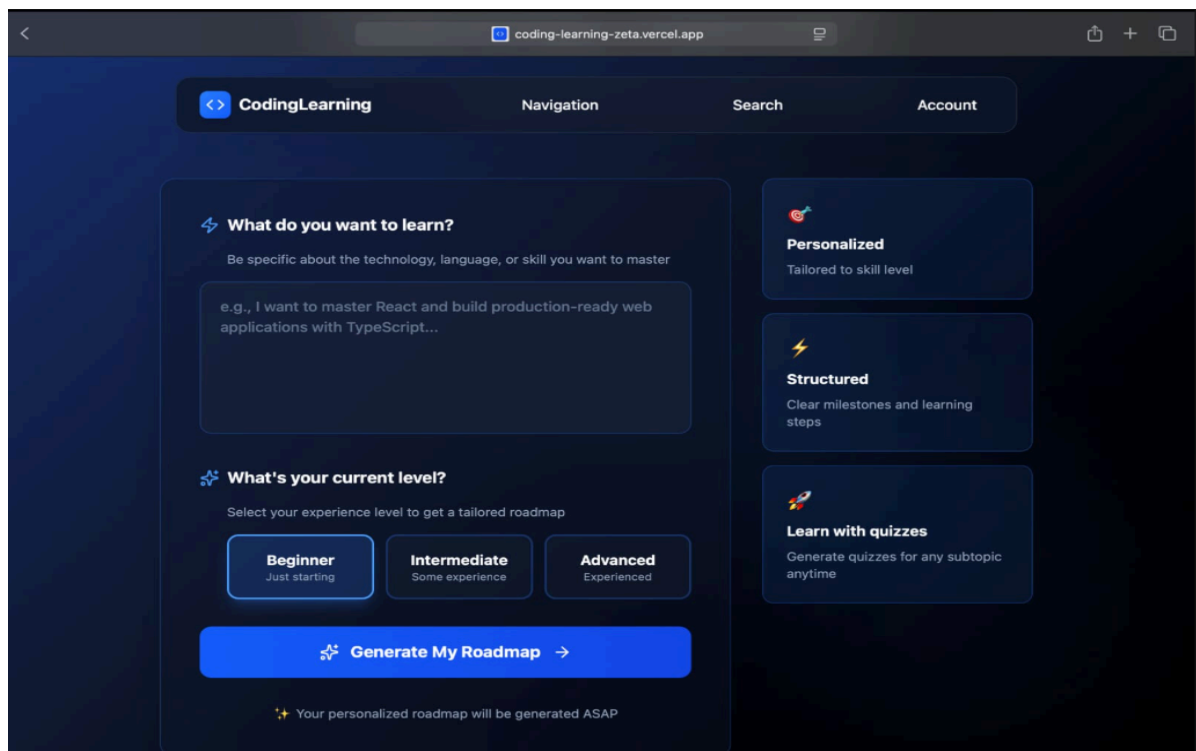
Chrome(Window):



Microsoft (Window):



Safari (IOS):



Mobile (Android) :

