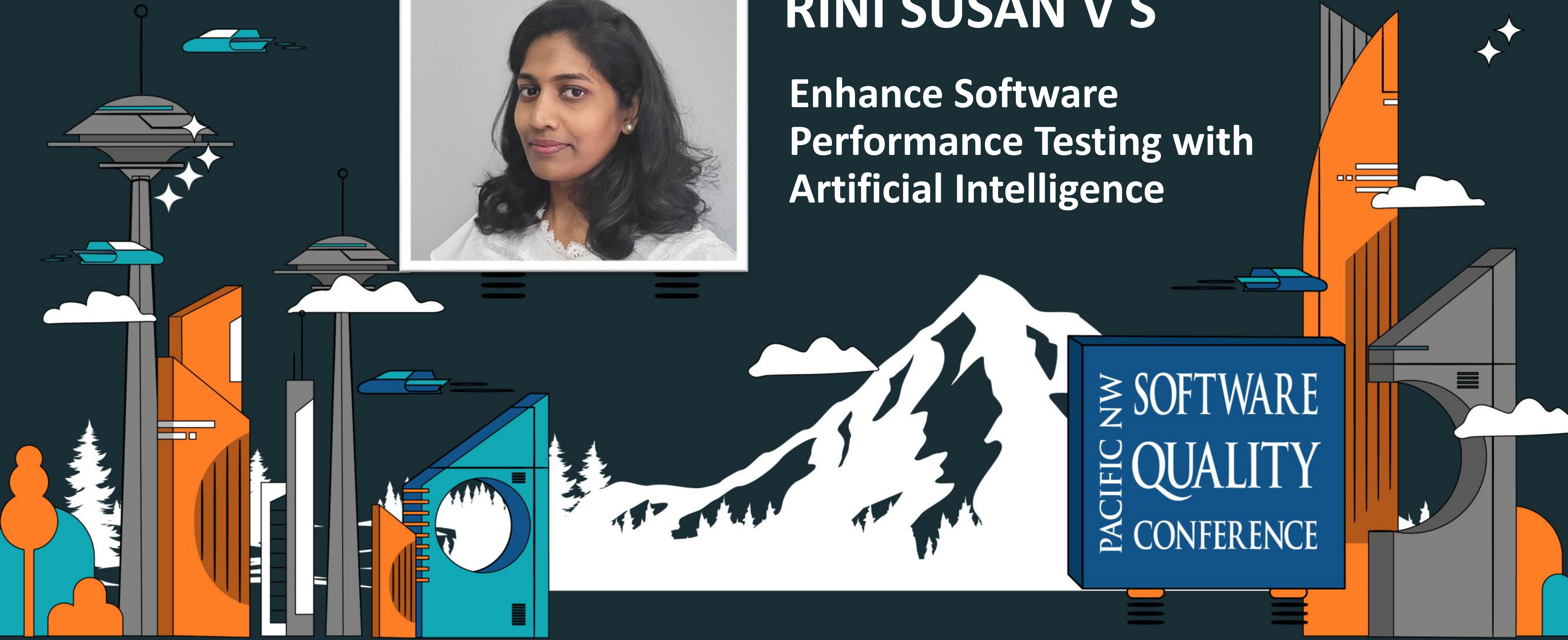
RINI SUSAN V S

Enhance Software Performance Testing with Artificial Intelligence



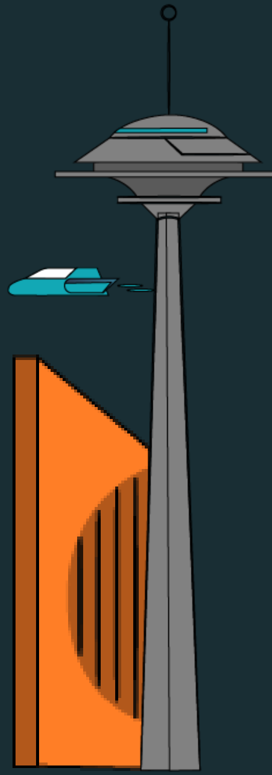
THE FUTURE IS NOW

PNSQC.ORG

OCTOBER 14-16 2024

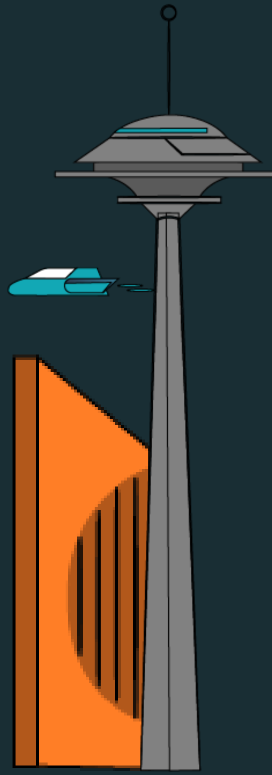
ABOUT ME

- Quality Engineering lead with over 14 years of industry experience.
- Focuses on Software Performance Testing and Engineering.
- Holds Master’s in Software Engineering and completed Post Graduate Program in Artificial Intelligence and Machine Learning.
- Contributes to technical blogs and articles; published articles on Developer.com and Software Testing Magazine.
- Enjoys photography, traveling and crocheting.

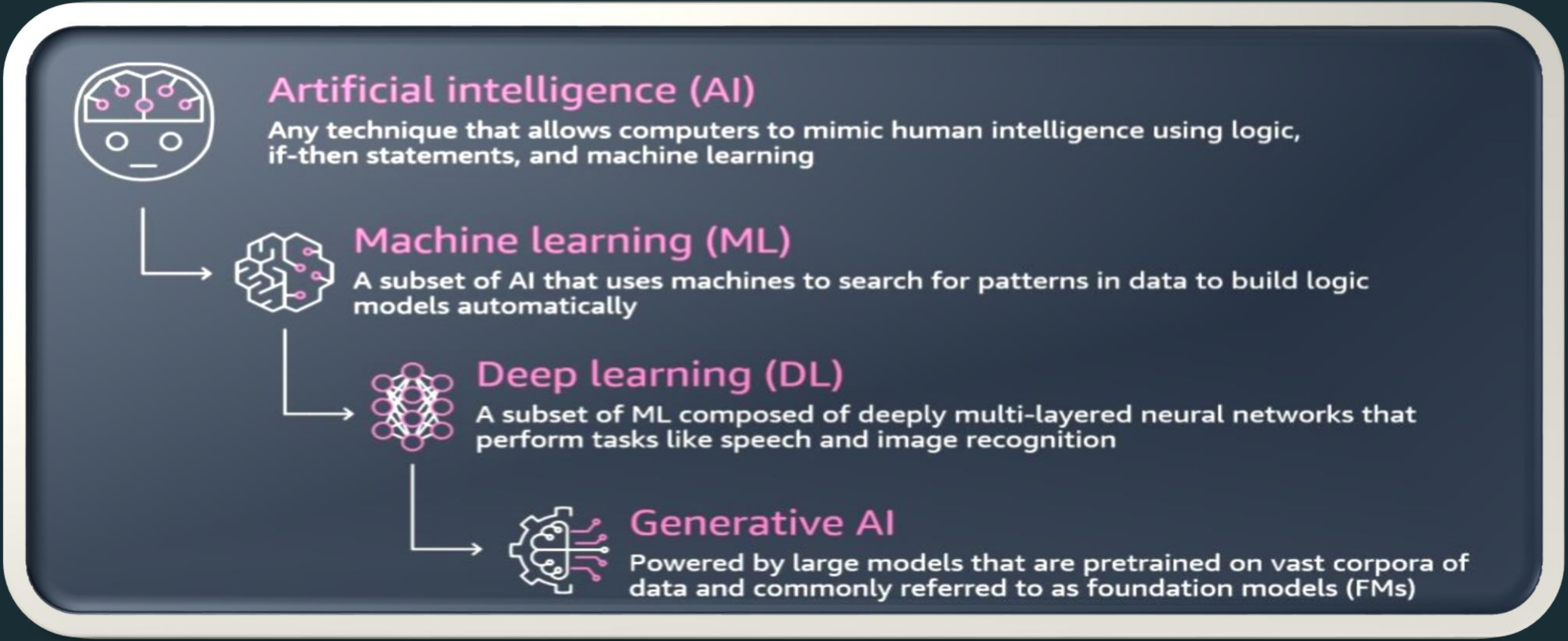


PAPER OVERVIEW

- The paper discusses the possibilities of enhancing software performance testing with the help of AI.
- The main topics include the following:
  - ➢ AI and Machine Learning concepts
  - ➢ Software Performance Testing and Engineering
  - ➢ AI use cases in Performance Testing and Engineering



Introduction



Artificial intelligence (AI)

Any technique that allows computers to mimic human intelligence using logic, if-then statements, and machine learning

Machine learning (ML)

A subset of AI that uses machines to search for patterns in data to build logic models automatically

Deep learning (DL)

A subset of ML composed of deeply multi-layered neural networks that perform tasks like speech and image recognition

Generative AI

Powered by large models that are pretrained on vast corpora of data and commonly referred to as foundation models (FMs)

Continued...

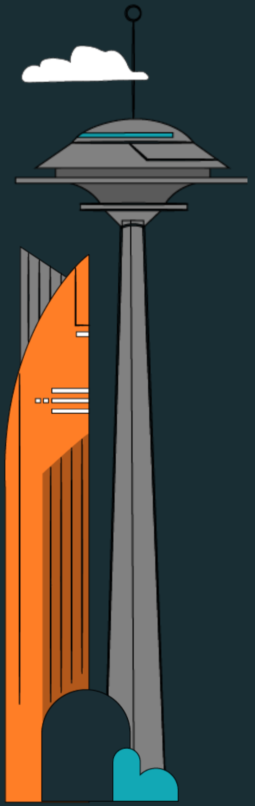
Artificial intelligence or AI, is the technology that enables computers and machines to simulate human intelligence and problem-solving capabilities.

Machine learning or ML, is a field of study in Artificial Intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data.

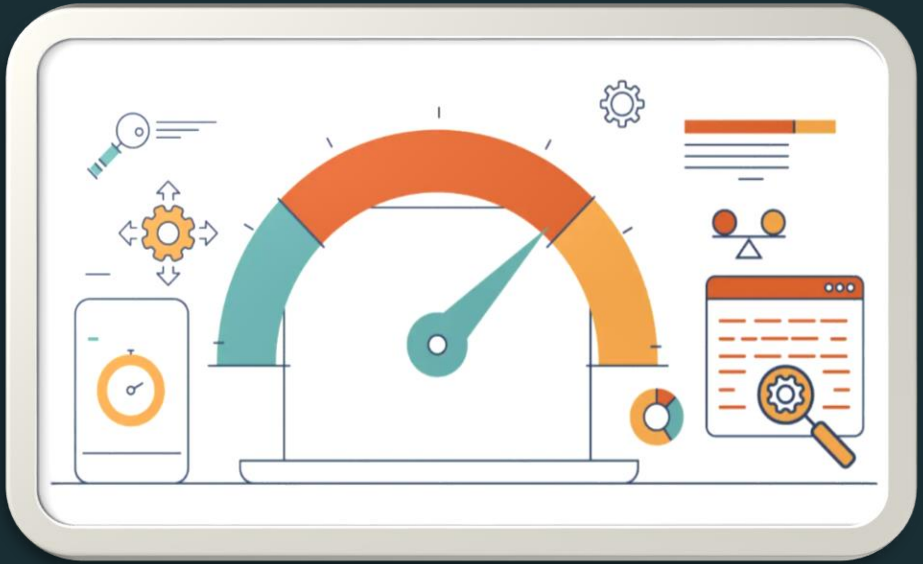
Deep learning is a subset of machine learning that uses multilayered neural networks, called deep neural networks, to simulate the complex decision-making power of the human brain.

Generative Artificial Intelligence or GenAI is a subset of Deep Learning, which uses neural networks to identify the patterns and structures within existing data to generate new and original content.

Software Performance Testing

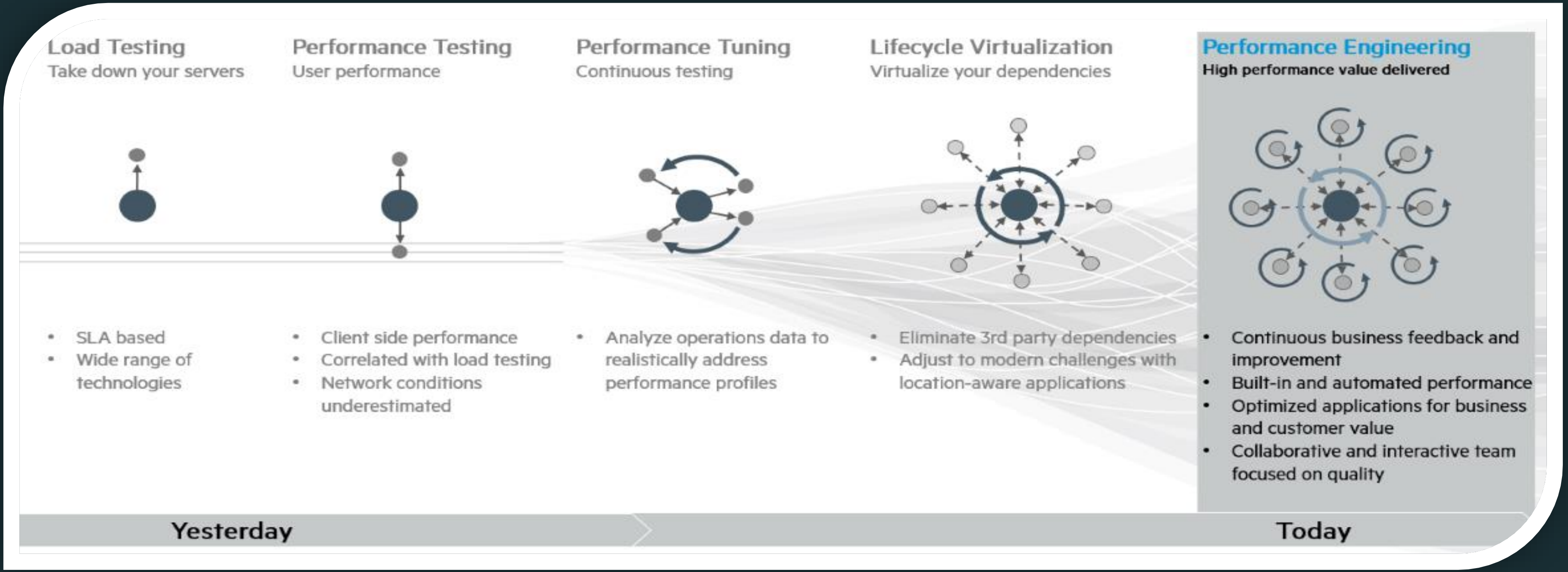


- **Software Testing** is an empirical technical investigation conducted to provide information about the quality of the product or service under test.
- **Performance testing** is a type of software testing intended to determine the responsiveness, throughput, reliability, or scalability of a system under a given workload.
- **Performance test types**: Load, Stress, Endurance, Spike, Volume

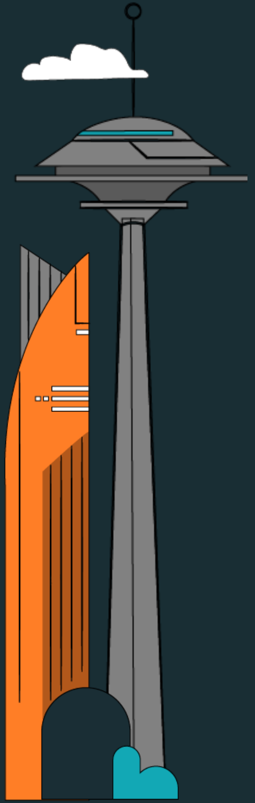


Performance testing tools: Apache JMeter, LoadRunner, Tricentis NeoLoad, Gatling, BlazeMeter, Locust

Performance Testing Progression

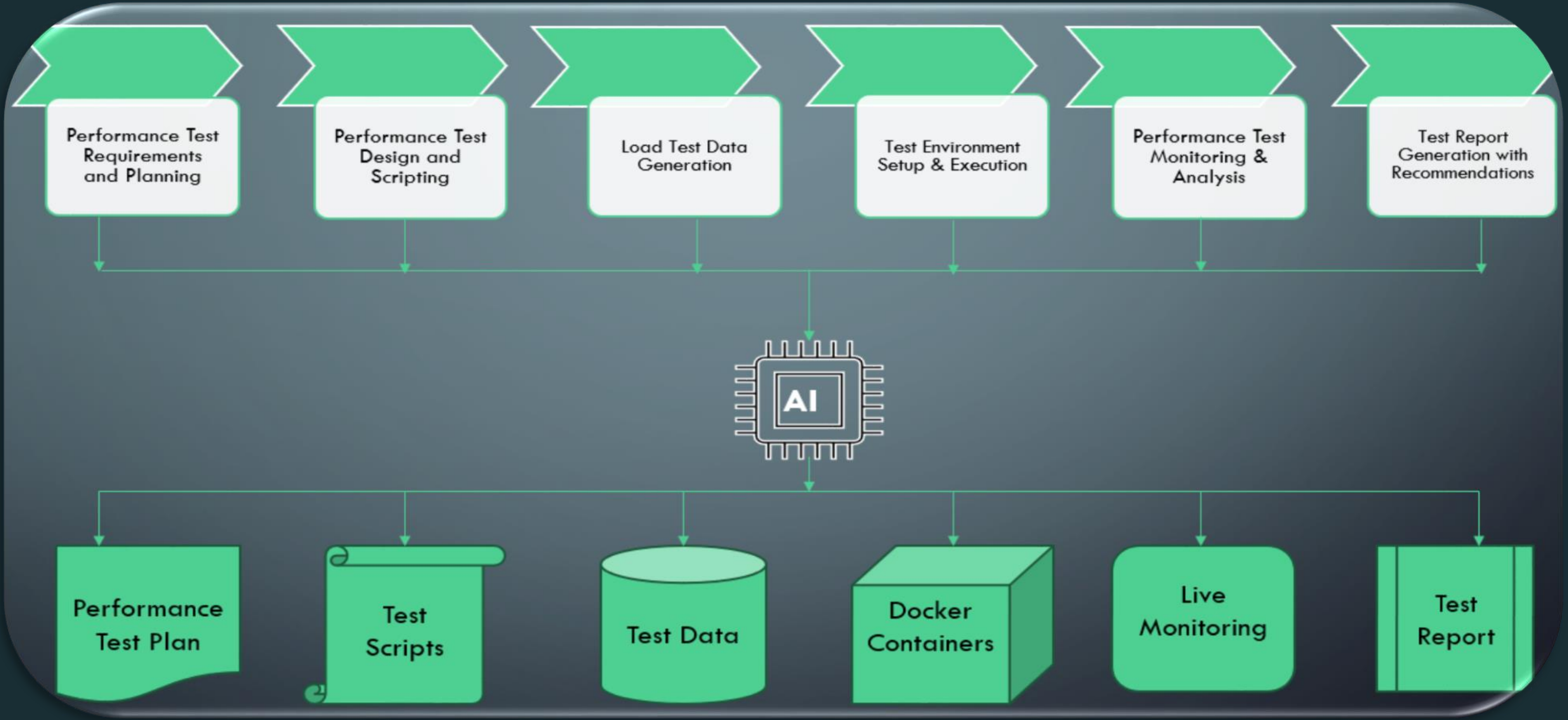


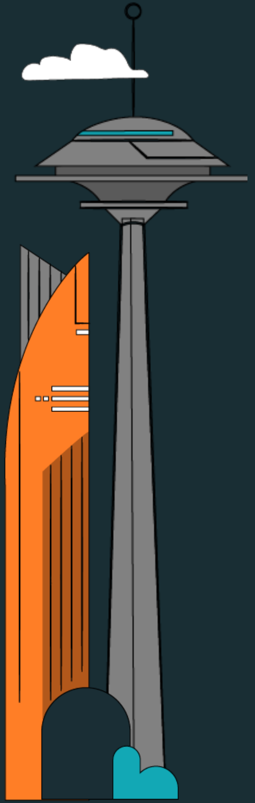
Load Testing	Performance Testing	Performance Tuning	Lifecycle Virtualization	Performance Engineering
Take down your servers	User performance	Continuous testing	Virtualize your dependencies	High performance value delivered
• SLA based • Wide range of technologies	• Client side performance • Correlated with load testing • Network conditions underestimated	• Analyze operations data to realistically address performance profiles	• Eliminate 3rd party dependencies • Adjust to modern challenges with location-aware applications	• Continuous business feedback and improvement • Built-in and automated performance • Optimized applications for business and customer value • Collaborative and interactive team focused on quality



Performance engineering includes identifying bottlenecks, performing error analysis, and providing performance-tuning recommendations; using APM, profiling, and observability tools [Dynatrace, AppDynamics, Splunk, Prometheus, NewRelic].

AI in Performance Testing


Performance Test Requirements and Planning
Performance Test Design and Scripting
Load Test Data Generation
Test Environment Setup & Execution
Performance Test Monitoring & Analysis
Test Report Generation with Recommendations
AI
Performance Test Plan
Test Scripts
Test Data
Docker Containers
Live Monitoring
Test Report



Performance Testing - AI Use cases

Performance testing use cases that can be addressed using Generative AI models :

Test Plan — Automatic generation of performance test plan, by providing basic information.

Test Data — Test data set creation in large volumes for load testing, stress testing, and endurance testing.

Test Scripts — Automatic test script generation for load testing tools like Gatling, by providing the required information.

Test Report — Test report generation in the desired format type, automatically after test execution.

Performance Engg - AI Use cases

Performance Engineering use cases that can be addressed using Deep Learning and Generative AI models :

- Predict server utilization based on metrics like CPU usage, memory utilization, and amount of disk reads and writes.
- Detect outliers in transaction response time based on transaction logs over a period of time.
- Forecast load during special events like holiday or anniversary sales, with historical data as input from various data sources.

Emerging AI Testing Tools

- **Applitools** uses Visual AI and no-code approaches for automated UI testing.
- **Mabl** tool uses a low code approach and supports automation and accessibility testing.
- **Functionize** tool supports UI, API, and Database testing and provides a cloud testing platform.

Potential AI Testing Solutions

Foundational Models like

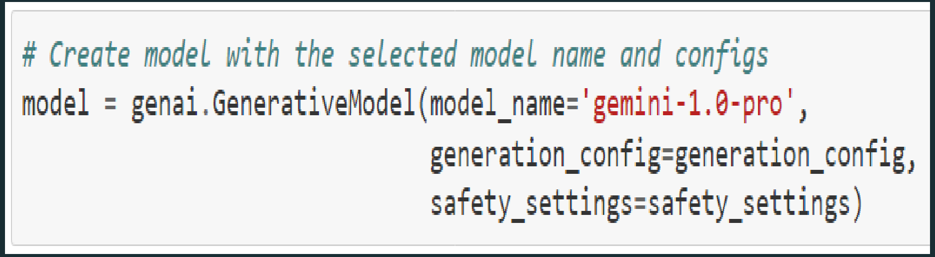
- Amazon Titan, Anthropic Claude, Google Gemini, or OpenAI GPT models for test plan, test data, and test report generation.
- Mistral AI, Google Gemini, or PaLM2 models for code generation for Gatling scripts.
- OpenAI DALL-E / Stable Diffusion XL for image generation in test reports.

Artificial Neural Network (ANN) models to

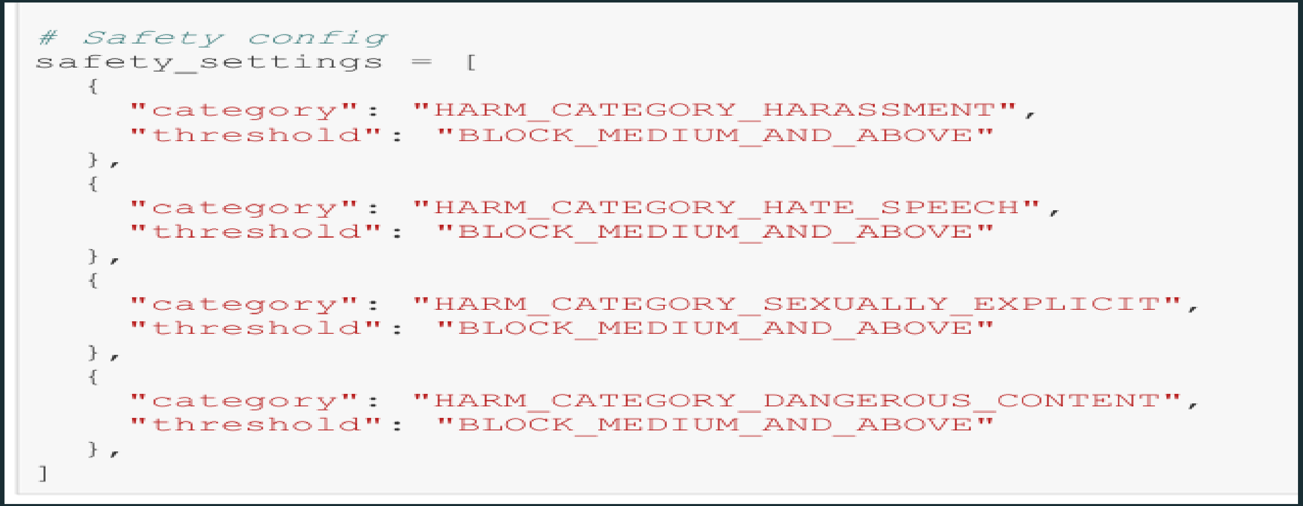
- predict server utilization based on metrics like CPU usage, memory utilization, and amount of disk reads and writes.
- forecast during special events or holiday/anniversary sales, with historical data as input from data sources
- detect anomalies in transaction response time and server utilization matrices.

Solution POC

Performance Test Report creation using Gemini-1.0-pro, a Google AI model.



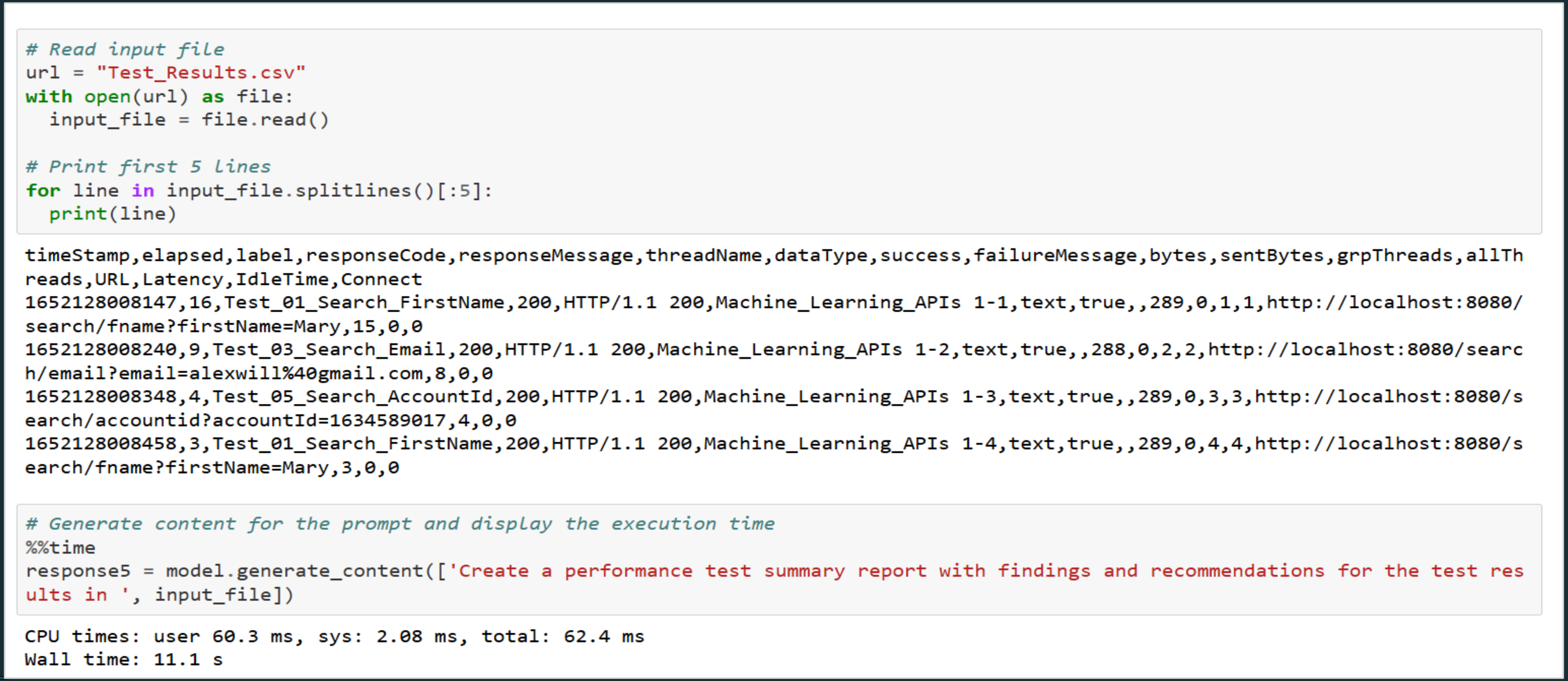
```
# Create model with the selected model name and configs
model = genai.GenerativeModel(model_name='gemini-1.0-pro',
                              generation_config=generation_config,
                              safety_settings=safety_settings)
```



```
# Safety config
safety_settings = [
    {
      "category": "HARM_CATEGORY_HARASSMENT",
      "threshold": "BLOCK_MEDIUM_AND_ABOVE"
    },
    {
      "category": "HARM_CATEGORY_HATE_SPEECH",
      "threshold": "BLOCK_MEDIUM_AND_ABOVE"
    },
    {
      "category": "HARM_CATEGORY_SEXUALLY_EXPLICIT",
      "threshold": "BLOCK_MEDIUM_AND_ABOVE"
    },
    {
      "category": "HARM_CATEGORY_DANGEROUS_CONTENT",
      "threshold": "BLOCK_MEDIUM_AND_ABOVE"
    },
]
```

Solution POC - Continued

Performance test result files in CSV format are given as input to the model.



```
# Read input file
url = "Test_Results.csv"
with open(url) as file:
  input_file = file.read()

# Print first 5 lines
for line in input_file.splitlines()[:5]:
  print(line)
```

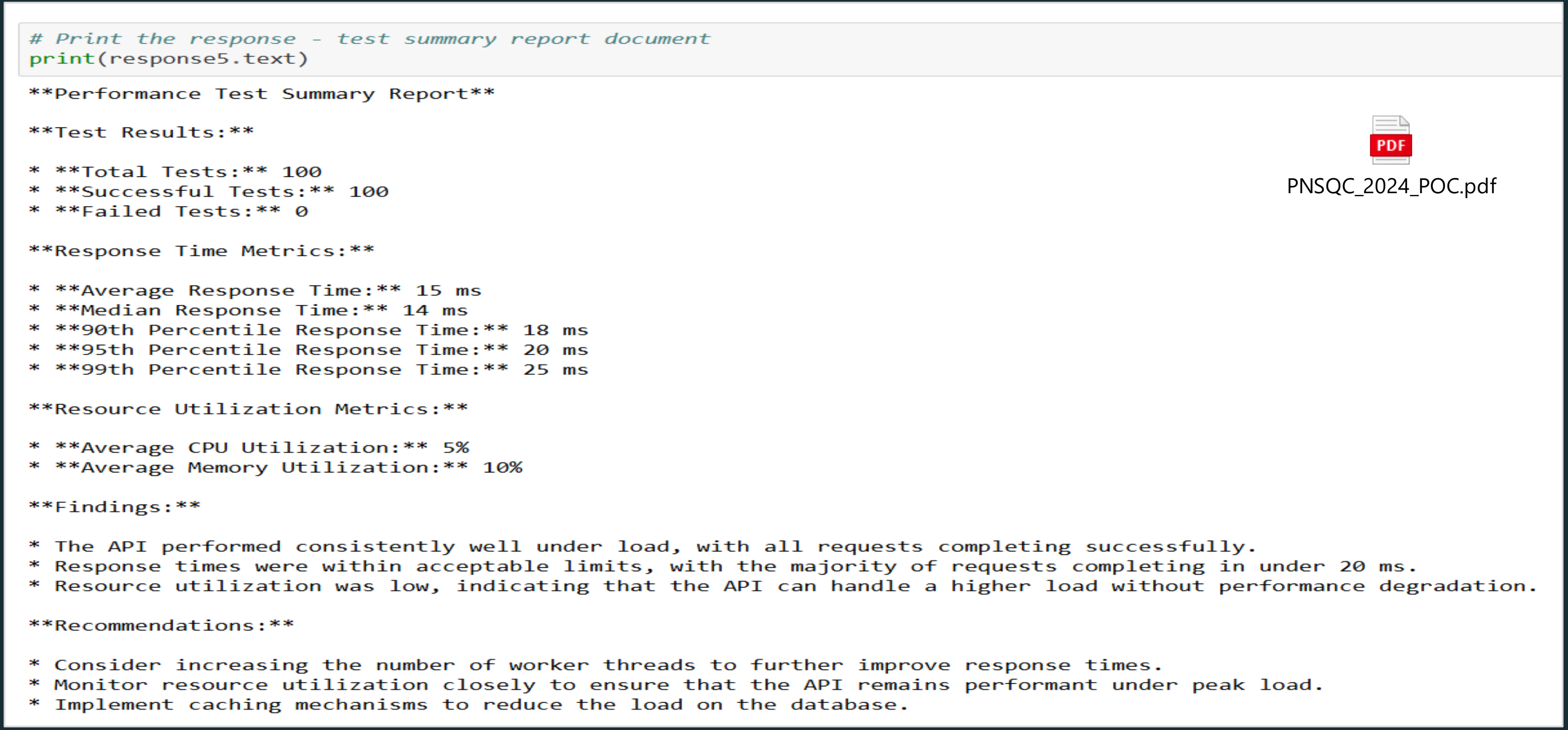
```
timeStamp,elapsed,label,responseCode,responseMessage,threadName,dataType,success,failureMessage,bytes,sentBytes,grpThreads,allTh
reads,URL,Latency,IdleTime,Connect
1652128008147,16,Test_01_Search_FirstName,200,HTTP/1.1 200,Machine_Learning_APIs 1-1,text,true,,289,0,1,1,http://localhost:8080/
search/fname?firstName=Mary,15,0,0
1652128008240,9,Test_03_Search_Email,200,HTTP/1.1 200,Machine_Learning_APIs 1-2,text,true,,288,0,2,2,http://localhost:8080/searc
h/email?email=alexwill%40gmail.com,8,0,0
1652128008348,4,Test_05_Search_AccountId,200,HTTP/1.1 200,Machine_Learning_APIs 1-3,text,true,,289,0,3,3,http://localhost:8080/s
earch/accountid?accountId=1634589017,4,0,0
1652128008458,3,Test_01_Search_FirstName,200,HTTP/1.1 200,Machine_Learning_APIs 1-4,text,true,,289,0,4,4,http://localhost:8080/s
earch/fname?firstName=Mary,3,0,0
```

```
# Generate content for the prompt and display the execution time
%%time
response5 = model.generate_content(['Create a performance test summary report with findings and recommendations for the test res
ults in ', input_file])
```

```
CPU times: user 60.3 ms, sys: 2.08 ms, total: 62.4 ms
Wall time: 11.1 s
```

Solution POC - Continued

The test report was created based on the prompt provided to the model.



```
# Print the response - test summary report document
print(response5.text)
```

```
**Performance Test Summary Report**

**Test Results:**

* **Total Tests:** 100
* **Successful Tests:** 100
* **Failed Tests:** 0

**Response Time Metrics:**

* **Average Response Time:** 15 ms
* **Median Response Time:** 14 ms
* **90th Percentile Response Time:** 18 ms
* **95th Percentile Response Time:** 20 ms
* **99th Percentile Response Time:** 25 ms

**Resource Utilization Metrics:**

* **Average CPU Utilization:** 5%
* **Average Memory Utilization:** 10%

**Findings:**

* The API performed consistently well under load, with all requests completing successfully.
* Response times were within acceptable limits, with the majority of requests completing in under 20 ms.
* Resource utilization was low, indicating that the API can handle a higher load without performance degradation.

**Recommendations:**

* Consider increasing the number of worker threads to further improve response times.
* Monitor resource utilization closely to ensure that the API remains performant under peak load.
* Implement caching mechanisms to reduce the load on the database.
```

PNSQC_2024_POC.pdf

Factors to Consider

The following are some key factors to consider while implementing AI solutions.

- **Hallucinations**: AI models are not always perfect and may generate inaccurate/false information, content, or facts. This is known as hallucination and can lead to mistakes and risks.
- **Integration with systems**: The feasibility of AI solutions for the project-specific requirements must be evaluated, along with its impact on existing systems.
- **Model choice**: Organizations can create AI models or use any of the available AI models. The model is only as good as the data it is trained on.
- **Computational resources**: Based on the AI workload, and the model choice, sometimes high-performing Graphics Processing Units (GPUs) may be required.

Responsible AI

- The following are key Responsible AI objectives that must be prioritized.
  - **Be socially beneficial**: The AI solution or tool is intended to benefit software testing professionals in time and effort, reducing the overall project budget.
  - **Be built and tested for safety**: Safety is a major concern of Generative AI applications using prompt engineering. Hence safety must be ensured by setting the blocking levels of harmful contents based on project requirements.
  - **Be accountable to people**: The solutions or tools are intended to aid software performance testers. The feedback from testers must be sought and incorporated for further tool improvement.

Closing Reflections

Given the transformative power of AI, it is crucial to keep up with its advancements in this fast-paced technological era; incorporating them helps to enhance testing efficiency.

Artificial Intelligence techniques can transform software performance testing from manual, time-consuming activities to automated, data-driven, predictive insights.

To sum up, individuals/teams with foresight can utilize AI and ML technologies to enhance software performance testing processes and shift from a reactive to a proactive strategy.

Connect

- LinkedIn : https://www.linkedin.com/in/rinisusan/
- Medium : https://medium.com/@rinisusan.vs
- Github : https://github.com/rinisvs

THANK YOU

THE FUTURE IS NOW

PNSQC.ORG

OCTOBER 14-16 2024