**Enoch Atuahene**

**Operating Systems: PM Lab (Web Server Case Study)**

**Reflection On A Web Server Handling Multiple Client Requests**

This lab was an eye-opening experience in web server performance testing. Setting up Apache, creating a simple web page, and then using Locust to simulate multiple client requests gave me a hands-on understanding of how web servers handle traffic. I found it particularly interesting to see the real-time impact of increased load on the system through the Locust web interface and the top command. The process of configuring Locust with the Python script to simulate different user behaviors (like visiting the homepage, product page, and cart) was intriguing and it helped me appreciate the complexity of real-world web traffic patterns.

The metrics provided by Locust and top were numerous and I didn’t get most of the acronyms, but going through the instructions sheet and as I watched them change in real-time, I began to understand their significance. The requests per second (RPS) metric in Locust shows how many users were hitting the server at any given moment, while the response times gave me insight into how well the server was handling the load. I found the failure rate particularly interesting, as it spiked due to the non-existent /product and /cart pages. On the top side, watching the CPU and memory usage fluctuate as the load increased helped me understand the relation between user requests and server resource utilization.

As the simulated load increased, I noticed several potential bottlenecks. The response times in Locust started to climb, indicating that the server was struggling to keep up with the incoming requests. The failure rate for the non-existent pages was consistently high, which in a real-world scenario could indicate missing content or routing issues. In the top display, I observed the Apache process consuming more CPU and memory as the load increased. This helped me understand that in a production environment, these could be indicators that the server might need more resources or optimization to handle higher traffic volumes efficiently.

This lab illustrated the importance of process management in a web server context. Each incoming request to the web server utilizes a process to handle it. As the number of requests increased, I could see through top how the system had to manage multiple Apache processes simultaneously, allocating CPU time and memory to each. This direct correlation between user requests and system processes demonstrated why efficient process management is crucial for web server performance.