**EEX5362**

**Performance Modelling**

**Mini Project**

Deliverable 1

T.D.K.L.C. Gunasekara

121426537

S9206657

# **System Overview**

The system I selected is a web server farm with a load balancer. These kinds of systems are commonly used in large-scale web applications such as e-commerce sites, streaming services, or social media platforms. It handles thousands of incoming user requests per second.

A load balancer acts as a controller that distributes incoming requests among several web servers. Each server processes user requests such as fetching pages, submitting forms, or loading media files. The purpose of the load balancer is to prevent any single server from becoming overloaded and to keep response times low.

This system was selected because it represents a real-world complex system that can be simulated and analyzed for performance. It includes multiple interacting components such as users, load balancer, servers, queues and also it has measurable performance parameters such as response time, throughput, server utilization, and queue length.

# **High-level Problem Description**

When many users access a website at the same time, the web servers can become overloaded. This causes slow response times, high queue lengths, and poor user experience. Without proper load balancing or scaling mechanisms, that system may fail to handle traffic peaks efficiently. Therefore, it is important to model and simulate the system to understand how performance changes under different traffic levels and server configurations. This helps to identify bottlenecks and find ways to improve the system’s stability and scalability.

# **Performance Objectives**

The main goals of this study are focused on improving system performance and reliability through the following objectives.

1. Minimize response time - To ensure that users receive quick responses even under moderate traffic.
2. Maximize throughput - To increase the number of requests processed per second without losing stability.
3. Avoid overloading servers - To keep server utilization below 90% to prevent performance degradation.
4. Optimize resource allocation - To use the minimum number of servers necessary to handle the current load efficiently.
5. Identify bottlenecks - To detect points in the system where requests start queuing up or delays increase sharply.