Performance Testing

Performance Testing Holiday Entry API Endpoint

Arthur Clarkson

**4th September 2024

Performance testing is a method used to assess the speed, responsiveness, and stability of a computer, network, software, or device under a specific workload. Organizations conduct performance tests to pinpoint performance-related bottlenecks.

The primary objective of performance testing is to detect and eliminate bottlenecks in software applications, ensuring optimal software quality. Without performance testing, systems may suffer from slow response times and inconsistent user experiences across the operating system (OS).

Within the functional specification, it was said that the create holidays entry API endpoint (`/Timesheet/CreateHolidayEntries`) was to respond in under 200ms for 100 requests at once. To test this API endpoint, it is work policy to create a python script to run, which will asynchronously send the HTTP requests at the same time, then take the average of all response times, if it is under time threshold the test is successful, with this knowledge I wrote this script:

```
api_url = "https://local.labman.co.uk:5002/Timesheet/CreateHolidayEntries"

# Function to send a single request and return response time
def send_request():
    start_time = time.time()
    try:
        response = requests.post(api_url, data)
        response_time = time.time() - start_time
        return response.status_code, response_time
    except requests.exceptions.RequestException as e:
        return None, None

# Function to perform the test
def performance_test(num_requests, max_workers=10):
    failed_requests = 0
    response_times = []

    # Using ThreadPoolExecutor to send concurrent requests
    with ThreadPoolExecutor(max_workers=max_workers) as executor:
        futures = [executor.submit(send_request) for _ in range(num_requests)]

        for future in futures:
            status_code, response_time = future.result()
            if status_code == 200:
                response_times.append(response_time)
            else:
                failed_requests += 1

    # Calculate average response time
    if response_times:
        avg_response_time = sum(response_times) / len(response_times)
    else:
        avg_response_time = 0

    # Print test results
```

```
    print(f"Total Requests: {num_requests}")
    print(f"Failed Requests: {failed_requests}")

    avg_response_time_ms = avg_response_time * 1000

    if avg_response_time_ms > 200:
        print("Test Failed, AVG Response Time Greater That 200ms")
    else:
        print("Test Passed, AVG Response Time Less Than 200ms")

    print(f"Average Response Time: {avg_response_time_ms:.2f}ms")

# Run the test with 100 concurrent requests
if __name__ == "__main__":
    num_requests = 100  # Number of requests to simulate
    performance_test(num_requests, max_workers=20)  # max_workers = number of concurrent threads
```

After running this script I get the following result, meaning it passed it's tests:

```
Total Requests: 100
Failed Requests: 0
Test Passed, AVG Response Time Less Than 200ms
Average Response Time: 131.55ms
```