Analysis: Service Downtime Scenario

- In this setup, I created three dummy service endpoints: /service1, /service2, and /service3. The /health route continuously monitors these endpoints by sending HTTP requests and recording their responses.
- When one service goes down — for example, /service3 returns a 500 Internal Server Error — the health check detects this in real-time. This detection works by:
- Checking the HTTP Status Code – If the response is not 200 OK (e.g., 500, 404, or no response), the service is marked as DOWN.
- Handling Timeouts or Connection Errors – If a request cannot connect within the set timeout (3 seconds in this case), it is treated as a downtime event.
- When a service is down, the /health endpoint will clearly show its status as DOWN along with the reason (error message or status code).

Detection in Production

In a real-world environment, this downtime detection could be enhanced by:

- Automated Monitoring Intervals – Running checks at fixed intervals (e.g., every 30 seconds).
- Alerting Mechanisms – Sending email or SMS when a service remains down for multiple consecutive checks.
- Historical Logging – Keeping records of uptime, downtime, latency, and error types for trend analysis and root cause investigation.
- This approach ensures that service outages are identified quickly, allowing developers or system administrators to take immediate action and reduce downtime impact.