1. Theoretical Understanding (30 min)

a. Explain I/O Bound vs CPU Bound

- Define each term and describe how performance bottlenecks differ.
- Give two real-world examples of I/O-bound workloads (e.g., microservices waiting on DB or API calls).
- How would you diagnose an I/O-bound bottleneck in production?

b. Concurrency & Parallelism

- Explain how asynchronous programming models (e.g., Go goroutines, Python async, Node event loop) mitigate I/O blocking.
- When should you use a worker pool vs event-driven model?

c. Scaling Strategies

- Describe vertical vs horizontal scaling and how each applies to I/O-bound services.
- Explain backpressure and circuit breaker patterns in high-throughput pipelines.

---

2. Practical Design Exercise (45–60 min)

Scenario

Your team is building a **chat analytics service**.
It must ingest messages from **50 000 concurrent chatbots**, process them via an external NLP API, and store summarized results in a database.
Each request to the NLP API takes ~200 ms on average.

Requirements

- Throughput target: **10 000 req/s**
- 99th percentile latency: **< 400 ms**
- Tolerant to partial external failures
- Scalable and observable

Tasks

1. **Design the architecture**: draw or describe components such as message queues, workers, load balancers, databases, and caching layers.

2. **Outline concurrency handling**:

   ○ How many workers/goroutines/threads?
   ○ How do you prevent resource exhaustion (connections, memory, etc.)?

3. **Explain fault tolerance**:

   ○ Retry, exponential backoff, circuit breakers, timeouts.

4. **Monitoring and metrics**:

   ○ What KPIs would you track to detect saturation?
   ○ How would you visualize throughput and latency?

5. **Write an optimized service**
   ○ Write the service in your preferred language does not need to be fully functional but the concept from

Deliverable

- High-level architecture diagram or textual breakdown.
- Example or pseudo code implementation in your preferred langauge

---

3. Optimization & Troubleshooting (15–30 min)

Given:

A service processes 5000 req/s with 80% spent waiting on HTTP I/O.
CPU usage: 10%
Memory usage: 65%
Average response time: 3 s

Questions:

- Identify the bottleneck and suggest 3 optimizations.
- Show how you'd measure improvement (benchmarks, profiling, tracing).
- Discuss trade-offs between concurrency and memory consumption.
- Write an optimized service in preferred language