Week 2 – Day 1: Backend Scheduler Architecture (Recall Notes)

1. What does “scheduler as separate server” actually mean?
It means running the scheduler in a completely separate Node.js process from the API. Not a different port or database — a different process lifecycle.

2. Why must scheduler not run inside API?
If API crashes, scheduler should continue. If scheduler crashes, API should continue. Time-based work must not depend on request traffic.

3. What is the role of the scheduler?
Scheduler is a clock, not a worker. It periodically checks the database for due work and triggers execution.

4. Why scheduler should never stop when no rows are found?
Because future work may become due even when no API requests arrive. Schedulers are time-driven, not event-driven.

5. Why database is used as a queue?
Database provides durability, atomic updates, and single source of truth. It safely coordinates multiple scheduler processes.

6. What problem does in-memory mutex solve?
It prevents overlapping scheduler ticks within the same process. It does NOT protect across multiple machines.

7. What ensures safety across multiple schedulers?
Atomic database update with condition: UPDATE ... WHERE status='PENDING'

8. How are simultaneous due calls handled?
Using ORDER BY next_action_at and LIMIT batching. Scheduler drains backlog gradually, not all at once.

9. Why API must not trigger execution directly?
APIs handle intent. Schedulers handle execution. Mixing them causes outages during deployments.

10. Why environment loading broke after separation?
Each Node process loads its own environment. dotenv reads from process working directory, not file location.

11. Why Postgres was not the problem?
Postgres was running, accepting connections, and accessible via psql. Failure was due to missing DATABASE_URL in Node runtime.

12. Key architectural rule learned
Schedulers are time-driven. APIs are request-driven. Database coordinates machines, not code.

13. What NOT to do
- Do not start/stop scheduler from API - Do not rely on in-memory retries - Do not introduce Redis/Kafka prematurely

14. Production mindset takeaway
Separation exposes hidden bugs early. Correct architecture feels boring when stable.

End of Week 2 – Day 1