Question 4: Manpower and Project Management
Problem Statement:
You are leading a team of 10 engineers working on a critical project with a tight deadline. Due to unexpected resignations, you are now down to 7 engineers. The project involves Node.js, Python FastAPI, PostgreSQL, and MongoDB. Outline a plan for how you would reallocate resources, adjust the project timeline, and ensure that critical milestones are still met.

Solution -

Re-evaluate Scope:

- Identify **must-have** vs **nice-to-have** features.
- Prioritize business-critical deliverables (MVP first).

Re-align with Stakeholders:

- Communicate transparently about resource loss.
- Present revised risk assessment and propose options: (a) trim features, (b) extend timeline, or (c) increase temporary support (contractors/consultants).

Skill Mapping of Remaining 7 Engineers:

- Who is strong in **Node.js** backend APIs?
- Who is strong in the Python **FastAPI & analytics pipeline**?
- Who can manage **DB optimization (Postgres + MongoDB)**?
- Who is flexible enough to cover multiple areas?

Original team = 10 → Now 7 (30% reduction).
Approach: Prioritize scope + overlap work with parallelism.

Phase 1 (Week 1–2): Stabilization

- Freeze **requirements**.
- Re-scope backlog (tag items: P0 = must, P1 = should, P2 = can drop).
- Assign ownership + backups.
- Establish daily syncs to detect bottlenecks early.

Phase 2 (Weeks 3–6): Core Delivery

- **Node.js team** delivers core APIs (P0).
- **FastAPI team** delivers analytics endpoints (P0).
- **DB Specialist** ensures schema & queries are production-ready.
- **DevOps** sets up CI/CD, infra monitoring, load testing environment.
- QA Engineer starts **parallel test automation** (instead of end-of-cycle).

Phase 3 (Weeks 7–8): Integration + Hardening

- API contract integration between Node.js + FastAPI.
- Stress test PostgreSQL queries + MongoDB transactions.
- Security review (auth, JWT validation).
- Bug bash + final regression.

Adjustments

- Drop/defer **low-priority dashboards/reports (P2)**.
- Delay **advanced optimizations** (e.g., caching layer, complex analytics) to post-MVP release.
- If timeline absolutely fixed → focus on **vertical slice delivery** (end-to-end core use cases working, not breadth).

Use **Agile sprint cycles (1 week)** with strict scope locks.
Define clear **milestones per sprint** (e.g., Node.js auth APIs done by Sprint 2).
Burn-down charts to visualize progress.
Continuous deployment to staging for early integration feedback.