1. Handling "important client" / "one more last" requests from the CEO

Don't just reject them outright, but don't let them bypass the process either. Run them through the same prioritization process that everyone else's requests go through, using the same criteria (value, cost, risk, effort). Then clearly show the tradeoff: "If we add this, here's what gets delayed or dropped." That turns it from a yes/no argument into a resource decision the CEO has to own.

If a request is genuinely urgent, utilize a well-defined exception or escalation protocol, such as a formal change-control process, rather than allowing informal alterations to the workflow. In the long term, engage the CEO in conversations regarding the consequences of recurrent reprioritization, including reduced team velocity and morale. Recognize that it remains the product manager’s responsibility to transparently communicate these effects, rather than absorbing them without commentary.

2. Hierarchical vs. matrix structure for a software support/service org

A. Hierarchical structure: Employees report to one supervisor aligned by expertise, such as development, QA, or network administration. This structure supports stable, high-volume tasks like tier-1 help desk and enables efficiency, clear accountability, and simple training. However, it can form operational silos if cross-functional expertise is needed quickly.
B. Matrix: People report to both a functional manager and a project/service manager. Good when support issues are cross-functional (an incident that needs dev, ops, and security together), flexible, and pulls the right people fast. Weakness: dual reporting creates conflicting priorities and slows decision-making.

In practice, combining these structures is often effective: use hierarchical models for routine, standardized support tasks, while applying a matrix approach when handling complex incidents or projects that require multiple skill sets quickly.