Recruting Challenge

Ziel: Baue ein kleines End-to-End-Feature oder einen Service, der für einen Sales-CRM/ERP-Kontext **spürbaren Nutzen** liefert.
Stack (minimal): Backend **Python**, Frontend **React**. Andere Stacks sind erlaubt.

Leitplanken (minimal)

- **End-to-End Interface:** Es gibt eine klar nutzbare Schnittstelle (UI, API **oder** CLI), über die ein\:e User\:in eine Aktion ausführt **und** deren Effekt in Daten **persistiert** und wieder **abrufbar** ist.
- **Persistenz:** einfache Speicherung (z. B. Datei, SQLite, Postgres o. ä.).
- **Mandantenbewusstsein (light):** Datenmodell mit `tenant_id` o. ä. + kurze Skizze zur Isolation.
- **README:** Problemwahl, Architektur-Skizze, Setup/Run, wichtigste Trade-offs, Next Steps.

Abgabe

- Repo-Link (oder ZIP)
- README (wie oben)
- Optional: kleines Deployment (z. B. Docker Compose/Cloudflare/Render)

Einschätzungsfragen

Im Rahmen der Challenge interessieren mich Deine Einschätzungen zu den folgenden Fragen:

1. **Tech-Stack:** Welchen Stack erachtest du im CRM/ERP Kontext am Sinnvollsten und wieso?
2. **End-to-End Interface:** Welche Variante (UI, API oder CLI) hast du gewählt und warum? Skizziere den Flow aus Sicht eines Sales-Users.
3. **Observability:** Welche strukturierten Logs/Metriken/Traces baust du ein? Was würdest du im Alerting überwachen?
4. **Frontend (falls UI):** Wie organisierst du State/Fetching (z. B. React Query), Fehler-/Loading-Zustände und ggf. Optimistic Updates?
5. **Performance & Skalierung:** Wo liegt im Prototyp der Engpass und wie würdest du ihn heben?
6. **Produktnutzen:** Welche Sales-Persona profitiert konkret und wie würdest du Impact messen ?
7. **Next Steps:** Was würdest du mit +1 weiterem Tag bauen und was bewusst *nicht*?